AI FOR MAPPING MULTI-LINGUAL RESEARCH PAPERS TO THE UNITED NATIONS' SUSTAINABLE DEVELOPMENT GOALS (SDGS)

Maurice Vanderfeesten
University Library
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
maurice.vanderfeesten@vu.nl
ORCID: 0000-0002-5119-3514

Robert Jaworek
Department of General Linguistics
Palacký University Olomouc
Olomouc, Czech Republic
robert.jaworek01@upol.cz
ORCID: 0000-0001-6419-640X

December 15, 2021

ABSTRACT

In this report we demonstrate how we made the multi-lingual text classifier to match research papers to the Sustainable Development Goals (SDGs) of the United Nations.

We trained the BERT multi-language model (mBERT) to classify the 169 individual SDG Targets, based on the English abstracts in the corpus of 1.4 million research papers we gathered using the Aurora SDG Query model v5.

This is a follow-up project of the query based Aurora SDG classification model. The purpose of this project is to try and tackle several issues: 1. to label also research output to an SDG that is written in a non-English language, 2. to include papers that use other terms than the exact terms used in keyword searches, 3. to have a classification model that works independent from any other database specific query language.

In this report we show how we decided to use the abstracts only and the mBERT model to train the classifier. Also we show why we trained 169 individual models, instead of 1 multi-label model, including the evaluation for prediction. We will show how to prepare the data for training, and how to run the code to train the models on multiple GPU cores. Next we show how to prepare the data for prediction and how to use the code to predict English and non-English texts. And finally we evaluate the model by reviewing a sample of non-English research papers, and provide some tips to increase the reliability of the predicted outcomes.

***K*eywords** Machine Learning · BERT · multi-lingual · SDGs · Sustainable Development Goals · United Nations · Scientific publications · Journal Articles · Mapping · Research Output

https://doi.org/10.5281/zenodo.5603019

1 Introduction

The Aurora Universities network is a network of nine leading European research universities founded in 2016, united by their commitment to match academic excellence with creating societal impact. University leadership asked the Aurora universities to demonstrate the relevance of their research to grand societal challenges [1, 2, 3]. Therefore the Sustainable development goals (SDGs) of the United Nations were chosen as a framework to be the leading narrative, to match research papers to these topics using boolean search queries [4]. The bibliometricians of the Aurora universities have worked together since 2017 to create our own definition of the *"Aurora SDG Queries"* matching our research to the SDGs, and narrow down to the level of the Targets within the Goals.

Based on an extensive feedback survey from 244 senior researchers across Europe [5], and a text analysis [6], this lead to a final version 5 of the *"Aurora SDG Queries"* in 2020 [7]. The quality of which these queries have been evaluated measuring the accuracy, we found an averaged Precision of 70% for all SDG's combined, and an averaged Recall of 14% [8].

Using the *"Aurora SDG Queries"* has an advantage that Boolean searches with keywords make it transparent how the search results came into existence for each one of the SDG-Targets. Still the drawback for many aurora universities was 1. *completeness of the source*, 2. *cross-platform actionability*, 3. *concept proximity*, 4. *multi-lingual content*.

1. *completeness of the source*; the Scopus database, on which the query syntax has been crafted, did not include all the research of a university. Scopus includes research papers only from selected journals. We need a solution or (combined) source(s) that is complete and can contain all research papers affiliated to a university. 2. *cross-platform actionability*; Crafting boolean searches are tailored to the syntax for a specific citation index database, Scopus. This makes it harder to use the queries in other systems like Pubmed, a CRIS system, repository, course guide, etc. We need a solution that is able to label research papers independently from a specific citation index. 3. *concept proximity*; the keywords used in queries produce a binary result set. If a term (eg. "hunger") is searched, and is in the metadata, then the paper show up in the result set. But if a term that is closely related (eg. "famine") that is mentioned in the metadata, but is not explicitly searched for, than it will not appear in the result set. This can lead to a large portion of research papers that are not related to an SDG, but in fact they might. We need a solution that is able to include research papers that contain concepts that are in semantic proximity. 4. *multi-lingual content*; Aurora university also produce research papers for local scientific and societal communities, often written in their national language. The Queries with English keywords do not work in other non-English languages, yet the research is still relevant to an SDG. We need a solution that is able to classify papers also in other languages.

This lead to the use of a pre-trained deep learning multi-language model, mBERT [9], which we will train the last layer to classify research papers to the SDG-Targets. This re-trained model will not solve the first issue of completeness of source, but we can present the model a text fragment in multiple languages from any source possible where it can recognise vectors of closely related concepts to determine it's class.

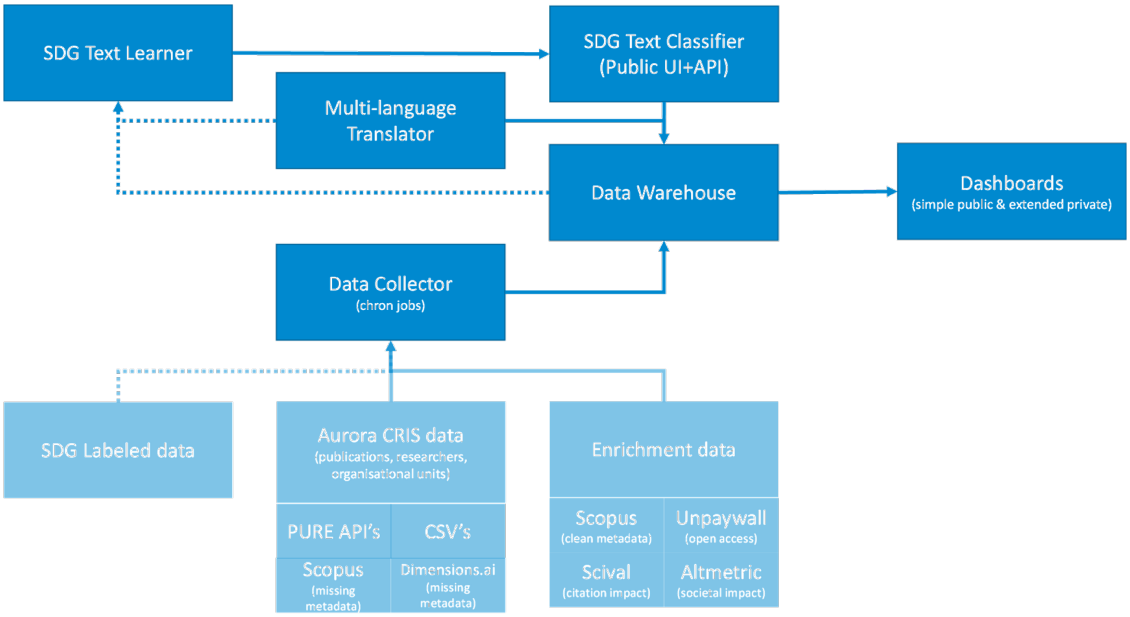

Figure 1: Workflow collecting data for training and prediction.

In this report we show how we decided to use the abstracts only and the mBERT model to train the classifier. Also we show why we trained 169 individual models, instead of 1 multi-label model, including the evaluation for prediction. We will show how to prepare the data for training, and how to run the code to train the models on multiple GPU cores. Next we show how to prepare the data for prediction and how to use the code to predict English and non-English texts. And finally we evaluate the model by reviewing a sample of non-English research papers, and provide some tips to increase the reliability of the predicted outcomes.

1.1 a short history of Natural Language Processing in Machine Learning

Most of us learned to read texts written in our mother tongue as children and we don't think much about this process later in life. We consider reading and speaking to be our natural skills. Only when we learn new languages, we remind ourselves that it is not so easy after all and certainly it takes some time. So how come that voice assistants in our cars and mobile phones speak almost as fluently as we do? Nowadays they can read poems or even tell fairy tales to our children. Search engines are capable of reading all the news and show us only these articles that would probably be most interesting to us, so we don't have to waist our time scrolling all newspaper websites every morning.

There's been a lot of research in linguistics, computer science and psychology – to name at least the most contributing fields – done before it became possible to model human language in the way it's used to help us in our daily life today. Most powerful language models are based on neural networks, algorithms inspired by our brain, which were discussed and proposed already in the 40s, but showed no good results then. They began to show their strength only with the arrival of the new millennium as the computing power of computers began to increase and the volume of digital data began to grow exponentially.

We can already spot a parallel here between human and machine learning. In both cases, to learn something new – in our case a natural language – takes time and effort, and at the same time data is needed for learning. When we learn for example Italian, we use a grammar textbook and a dictionary which were improved to their current form over the past decades. For a long time, natural language processing also drew a lot from linguistic rules derived from the textbooks. But nowadays we don't use rule-based language modeling anymore. Instead, we use statistical models capable of deriving the laws of natural language only from raw texts. We have to feed them a huge amount of text data, which we actually need to modify a bit before that.

1.2 Why we use the BERT language model

In Aurora Research Dashboard we use a variation of a BERT language model that was introduced by Google in 2018. It is basically a pretty big neural network capable of context learning. It was pre-trained on a large corpus of unlabeled text including also the entire English Wikipedia. As a result, it is relatively easy today to train him to handle a specific task, in our case document classification.

In a short investigatory study by students at the Vrije Universiteit Amsterdam in the Text Mining Domains Course, they tested what features of the metadata give the most predictive value, with what BERT model, mBERT or XML-RoBERTa.

See Github for the code `https://github.com/Aurora-Network-Global/TMD`

They found that, of several tested configurations, a system with the multilingual BERT (mBERT) model and the abstracts of scientific publications as input achieved a macro f1-score of 0.912. These results can hopefully form a basis for further research in the field of SDG classification. [10]

Name	Model	Input	Max Tokens
mBERT-t	mBERT	titles	128
mBERT-a	mBERT	abstracts	400
mBERT-ta	mBERT	titles, abstracts	400
XLMR-t	XLM-RoBERTa	titles	75
XLMR-a	XLM-RoBERTa	abstrats	300

(a) Overview of the different test configurations.

	precision	recall	f1-score
baseline	0.346	0.344	0.322
mBERT-t	0.828	0.824	0.826
mBERT-a	**0.916**	**0.911**	**0.912**
mBERT-ta	0.905	0.893	0.899
XLMR-t	0.783	0.704	0.734
XLMR-a	0.799	0.829	0.810

(b) Performance based on macro scores for all test configurations.

Figure 2: Model configurations tested, and test scores

2 Data preparation for training

In this section we explain how we prepared the data for training. This explains:

1. Provenance; the origin of the data. How we collected the data for each of the targets as true positives.
2. Transformation and data format for training
3. Method for normalisation of sample sizes and selection of true negatives
4. Distribution percentages of the training set, validation set and test set
5. Preprocessing the abstracts

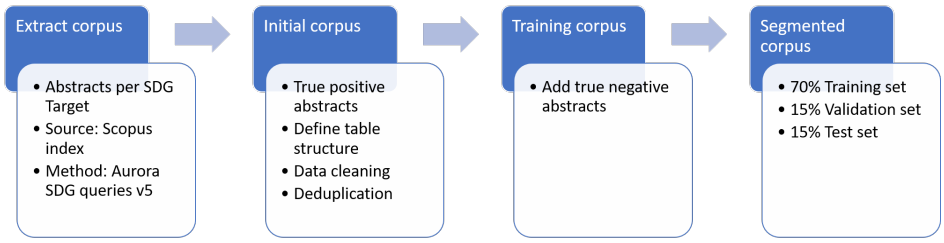

Figure 3: Steps for data preparation for training.

Provenance; the origin of the data. How we collected the data for each of the targets as true positives. We used Scopus as our source. For each SDG Target we ran a query using the Aurora SDG queries version 5, where we extracted the identifier and the abstracts from the resulting publications metadata.

Those queries can be found here: `https://aurora-network-global.github.io/sdg-queries/` . Please note that the average precison of these queries is 70% and the recall 14%, according to a panel of 244 researchers. Find the evaluation report here: `https://doi.org/10.5281/zenodo.4964606` This precision also reflect the outcomes of the trained classifier.

We uploaded the original data here: `https://doi.org/10.5281/zenodo.5224005`

To get a grasp of the amount of data we used for training each sdg target, we've put below some graphs and statistics.

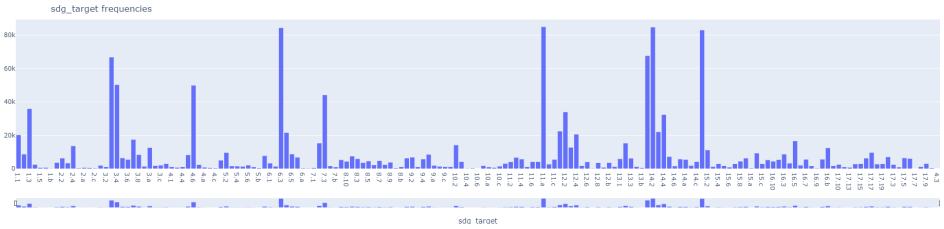

Figure 4: Number of abstracts per SDG target.

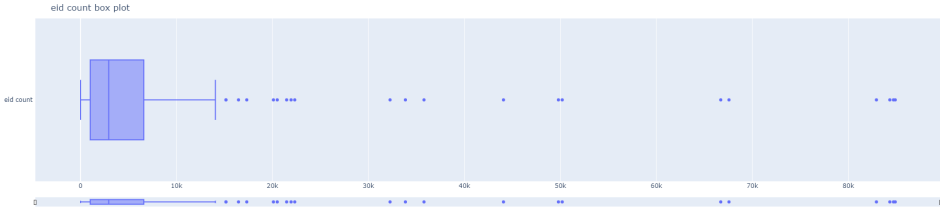

Figure 5: Distribution of sample set sizes of abstracts per SDG target

The sample sizes of most sdg target sets contain between around 1000 and 6500 abstracts.

The smallest set is sdg target 10.6 with 11 abstracts, followed by sdg target 4.3 with 27 abstracts. The biggest set is sdg target 11.a with 84890 abstracts, followed by sdg target 14.2 with 84702 abstracts. The median is sdg target 11.1 with 2937 abstracts

Initial Corpus - Data Cleaning, Transformation and Structure for training Our training data is stored in a table containing 170 columns. In the first one there are abstracts of scientific papers written in English, in the rest of them are zeros and ones. If an abstract belongs to a specific SDG target, there's a one (1) in the appropriate column. And there are zeros (0) in the rest of them.

abstract	1.1	1.2	...	17.16	17.17
Lorem ipsum dolor sit amet, consectetuer adipiscing elit.	0	1	...	0	0
Aenean commodo ligula eget dolor.	1	0	...	1	0

Table 1: Example of the data structure the initial corpus has: the abstracts appearing ("1") in each of the queries related to the SDG targets.

After that we clean the abstracts from leading and trailing copyright information Scopus attaches from publishers. We use this regular expressions:

```
(\©(.*\.|.*(...)))|(\©\d{1,4}\.)
```

The table contains duplicate abstracts, but they can appear in multiple sdg targets. We appended the 1's of the other sdg targets to the first appearing abstract. and removed the others rows. This helps better define an opposing negative class for training purposes.

Training Corpus - Method for normalisation of sample sizes and selection of true negatives For a neural net to train for each class (sdg target) equally, we have to feed it the same number of abstracts. If we were to build one multi-class model, we need to find the sdg target with the lowers amount of abstracts, and reduce the number of abstracts for all other sdg targets. This low sample size is not favourable since no clear features can be trained, plus we miss a negative class, that does not belong to that target.

Instead we trained 169 models, one for each sdg target. We defined the abstract from that sdg target as the true positive class, and an equal amount of abstracts representing the true negative class.

We selected the true negative abstracts by the following procedure:

1. Select an sdg target and create a new table to add true negative abstracts
2. Count the number of abstracts x of the selected SDG target.
3. And Randomly select the number x of abstracts we need to fetch from the other SDG target classes.

Segmented Corpus - Distribution percentages of the training set, validation set and test set When training a machine learning model, we also want to see how well the model has performed. So we separate the training corpus into 3 segments by randomly selecting the equal amount of true positive and true negative abstracts. 70% in the training set, 15% in the validation set, and the other 15% in the test set.

Preprocessing the abstracts Further, abstracts need to be preprocessed through model specific tokeniser. BERT accepts text input in the maximum length of 512 tokens. To those which are already in the text are added also special tokens [CLS] and [SEP], which indicate start and end of each sentence respectively.

```
['[CLS]', 'since', 'october', '2006', 'the', 'specialist','training', 'for',
'house', 'officers',  'in', 'psychiatry', 'in', 'the', 'province', 'of',
'north', 'holland', 'has', 'started', 'with', 'a', 'one', '-', 'week',
'intensive', 'ind', '##uction', 'course', '.', '[SEP]', 'the', 'course',
'has', 'been', 'designed', 'to', 'facilitate', 'the', 'transition', 'to',
'specialist', 'training', 'and', 'to', 'provide', 'train', '##ees', 'with',
'the', 'tools', 'they', 'need', 'for', 'dealing', 'with', 'psychiatric',
'emerge', '##ncies', '.', '[SEP]', 'fifteen', 'house', 'officers', ',',
'divided', 'into', 'three', 'focus', 'groups', ',', 'were', 'asked', 'for',
'their', 'views', 'on', 'the', 'course', '.', '[SEP]', 'they', 'gave', 'the',
'course', 'a', 'high', 'rating', 'and', 'stated', 'that', 'it', 'had', 'had',
'a', 'positive', 'effect', 'on', 'group', '-', 'bond', '##ing',
'and','function', '##ing', 'in', 'their', 'subsequent', 'training', '[SEP]']
```

Figure 6: Example of a tokenised abstract.

After the tokenisation abstracts longer than the limit need to be truncated. At this point we still have abstracts in the form readable for humans but not a computer. Now we need to convert text to numbers, i.e. word embeddings. Word embeddings are dense vector representations of words in lower dimensional space.

By translating a word to an embedding, it becomes possible to model the semantic importance of a word in a numeric form and thus perform mathematical operations on it. As we said before, BERT is capable of context learning which

happens in the training. It means that the model is e.g. able to recognize that the word "bench" has a different meaning in two following sentences:

"I lift 100 kg on the bench press."

"I like to read on a park bench."

In both cases the word "bench" has different vector representation, which is quite helpful in the text classification tasks as ours. After the tokens are converted to embeddings, we can use abstracts and their corresponding labels (0s and 1s based on the targets) as input to our classification model.

3 Model building

In this section we describe why we chose to re-train an existing pre-trained model and why we chose to work with many bi-label models instead of one multi-label model.

Re-training an existing pre-trained model Once we have pre-processed our data, divided it into a training set, validation set and test set, it's time to build the model. There are several ways to do that. We can build the whole neural network from scratch. This requires advanced coding skills and takes a lot of time and effort – and professional developers rarely take this path nowadays. Far more common is the so-called transfer learning. In this case, we take a model pre-trained on a general data/task and modify it further according to our specific needs. A great place to download pre-trained models is the open-source platform Hugging Face.

There are many state-of-the-art machine learning models invented by companies like Google or Facebook, which can be used for image or natural language processing etc. Anyone can download them for free and learn how to use them thanks to detailed instructions. We chose a multilingual variant of the Google BERT model, the so-called mBERT model.

After downloading the selected pre-trained mBERT model, it's possible to write the rest of the pipeline using Tensorflow or Pytorch libraries. These are currently the two most used libraries for deep learning. But there is also a third option, namely Keras. Keras is a deep learning API running on top of Tensorflow. It eliminates a lot of coding because it is written to make building models as efficient as possible and it is compatible with both Pytorch and Tensorflow. Once we have already finished pre-processing of our data, the model itself could fit in only a few lines of code.

Here will be the code:

`https://github.com/Aurora-Network-Global/sdgs_many_berts/blob/main/train.py`

As we can see, the whole model implementation is pretty quick. Firstly, we import the mBERT model from Hugging Face. Secondly, we configure so that it is suitable for our data. Since every model takes some inputs and return outputs, i.e. predictions, we have to add an input and output/target layers and specify their attributes. In our case, inputs are tokenised abstracts in the maximum length of 512 tokens. If there are any longer ones, they are shortened; the shorter ones on the other hand are padded with zeros.

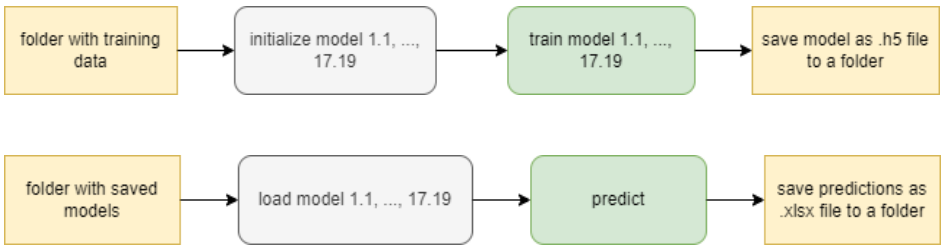

Figure 7: The architecture how the model build is orchestrated.

Model Implementation Architecture We want our model architecture to predict if a paper reflects zero or more of the Sustainable Development Goals Targets. There are two ways to that. We can train one multi-label classification model or multiple smaller ones and then join them together. The second approach has a few benefits over the first one. By training model for each target, we can fine-tune each one of these models separately. Since we don't have the same number of training data per target, training one multi-label model would require complicated calculations of target's weights. However, when choosing the second way, we can generate balanced training corpus for each target by

randomly picking exact same number of abstracts from all the data as the target we are currently working on has. When the training is done, we have 169 separate binary classifiers, which predict whether a paper belongs to the specified target or not. By concatenating them together, we get one big multi-label classification architecture.

4 Training and evaluation

In this section we show how to run the code to train the models on multiple GPU cores, including the evaluation for prediction.

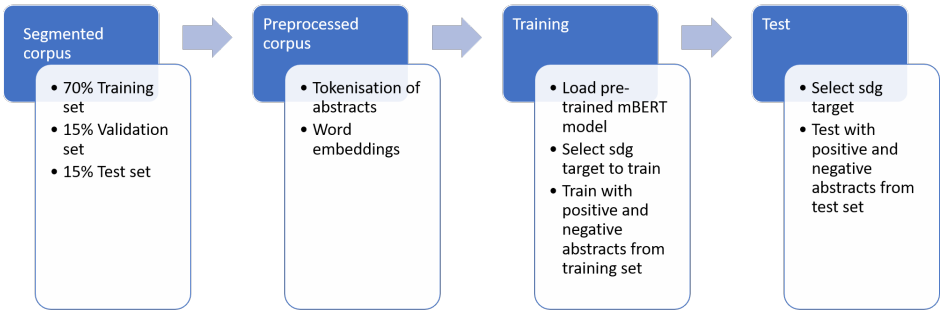

Figure 8: Steps for training and testing.

Training Once we have pre-processed the training data a written the code, it's time for a training itself. Since we use pre-trained mBERT models for our task's needs, we don't need as much time as if we had to train models from scratch. But still, it takes not negligible number of hours. To speed up the process a little bit, we used GPU. It is nowadays standard to use GPUs rather than CPUs when working with deep learning architectures because GPUs cope better with large number of simple computations. These simple computations are setting weights of nodes in the network – in each step of the training they change its value until the network learns to detect desired statistical patterns in the data.

It is standard procedure to split the training data into three separate sets: train, validation, and test. The first two are used while training for setting model weights and their correction. The third one – which consists of data model has never seen before – is used for testing. This way we make sure our model learned what we wanted. If it scores good also on test set, it means it generalize well and did not overfit training data.

Since our problem is the nature of the multiclass-multilabel classification, we decided to simplify it by training 169 binary classification models and then merge them into one big architecture. The train script uses as many models as there are classes in a training set. It overcomes a problem of imbalanced classes in multiclass-multilabel classification by slicing the training data into smaller sets based on individual classes in the following manner: select all instances belonging to class 1; select randomly from the remaining classes the same number of instances as in class 1 to represent class 0. The scripts were written to classify SDG targets.

Test evaluation statistics Statistics show that the 169 trained models have high levels of accuracy (on average above 95%) and low levels of losses. Meaning it does a pretty good job on labeling a text to an sdg target, or as a non-sdg-target.

target	accuracy	precision	recall	f1 score
1.1	0,956	0,957	0,955	0,956
1.2	0,963	0,980	0,946	0,963
...				
17.18	0,912	0,878	0,949	0,912
17.19	0,926	0,924	0,933	0,929

Table 2: Example test statistics returned.

For a complete list of statistics, download this table: `https://zenodo.org/record/5761472/files/SDGs_many_BERTs_models_test_statistics.csv`

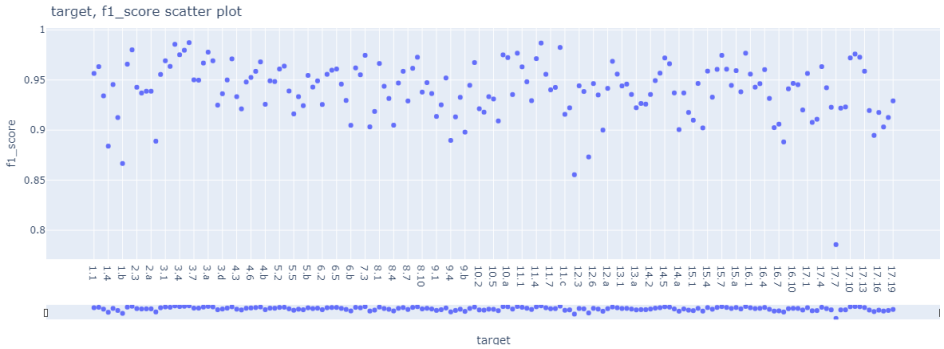

Figure 9: Results of the F1-scores for each of the sdg target models.

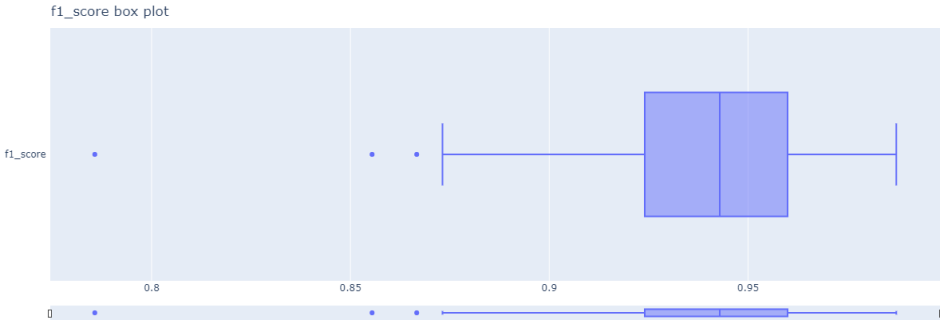

Figure 10: Distribution of the F1-scores of the sdg target models.

The majority of the sdg target models have an F1-score between 0.92 and 0.96. The median is 0.94. the minimum is 0.78 (sdg target 17.7), the maximum 0.98.

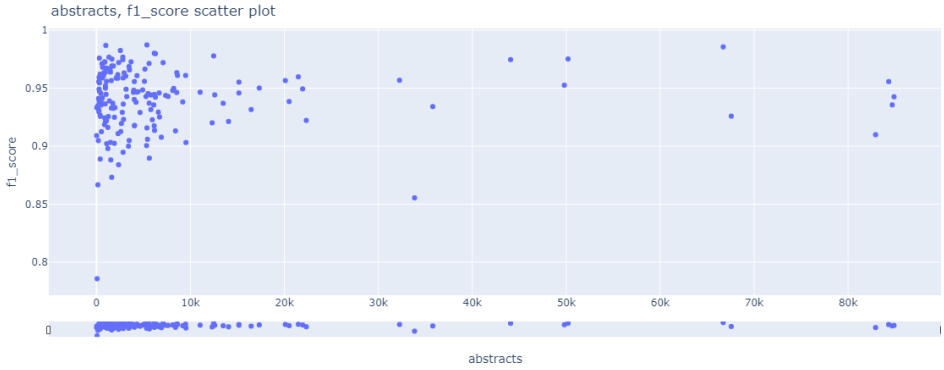

Figure 11: F1-scores vs Abstract sample sizes

In figure 11 we see that the bigger the sample set of the abstracts, the better the f1 score will be, but not in all cases.

5 Data preparation for prediction

For prediction, we must first download the relevant models from Zenodo repository. They are divided into groups according to individual SDGs. This way, everyone can decide whether (s)he wants to use all of them or focus on just

selected ones. For a complete list of download urls, start by downloading this table: `https://zenodo.org/record/5761472/files/SDGs_many_BERTs_models_download_urls.csv`

Once the we have the models saved on our local machine, it's time to get the texts we want to classify. It doesn't matter in what format they are saved, as long as we are able to load them in python console. We just need to convert the texts into a list (python) before feeding them as input to the prediction script. The pre-processing is for the prediction the same as for the training and everything will be done automatically. There is nothing to be changed if using non-English texts. To get you started, you can download this list of abstracts: `https://zenodo.org/record/5761472/files/predict_data_sample_multiple_language_abstracts.csv`

6 Prediction

After we downloaded the models and loaded the data we want to classify into the python console, we just run the prediction script which could be found here: `https://github.com/Aurora-Network-Global/sdgs_many_berts/blob/main/predict.py`

First of all, we need to check if we have installed all the dependencies. Second, we specify MODELS and SAVE_PREDICTIONS_TO variables, which correspond to the path to the folder on your machine where you downloaded the models and to the folder on your machine where you want prediction scores to be saved, respectively. Then we set DATA variable to the list of texts we loaded. Afterwards we could just run the script and wait for it to process the data.

mBERT model outputs a probability score saying how confident the model is each text belongs to a target. The output of the script is an excel file with text IDs and their scores per each target. We recommend to set a threshold for these scores to be 0.95 for English texts and even higher for non-English ones. We can filter predictions above specified threshold either in the python console using function predictions_above_threshold(predictions_dataframe, threshold=0.95) or later on semi-manually in the excel file.

7 Evaluation

We Evaluate to provide an answer to the assumptions stipulated above.

1. The AI models are able to label also research output to an SDG that is written in a non-English language,

We will evaluate the models by reviewing a sample of English and non-English (Dutch, German, Czech) research papers. We assume that models will perform differently on shorter and longer texts. Therefore we will evaluate them on selected groups of abstracts based on their length in tokens from BERT vocabulary. The groups/intervals are: <1,100>, <101, 200>, <201, 300>, <301, 400>, <401, 512>.

TO BE CONTINUED

2. The AI models include papers that use other terms than the exact terms used in keyword searches.

We demonstrate this by using the number of terms / tokens used by the query for sdg target x.x (equally weighted), and compare the is to the terms / tokens used by model for sdg target x.x. (weighted differently)

TO BE CONTINUED

3. With the AI models we can classify papers independent from any other database specific query language.

We demonstrate this by referring to point 1, where we are able to label papers from a spreadsheet, instead of a external database.

8 Conclusion

Concluding answering the assumptions: 1. to label also research output to an SDG that is written in a non-English language, 2. to include papers that use other terms than the exact terms used in keyword searches, 3. to have a classification model that works independent from any other database specific query language.

To be written when evaluation on non-English texts is done.

TO BE CONTINUED

References

[1] Maurice Vanderfeesten and René Otten. Societal relevant impact : Potential analysis for aurora-network university leaders to strengthen collaboration on societal challenges. Aurora-Network Norwich 2017 (Aurora2017).

[2] L. van Drooge, P. van den Besselaar, G. M. F. Elsen, M. de Haas, J. J. van den Heuvel, H. Maassen van den Brink, B. van der Meulen, J. B. Spaapen, and R. Westenbrink. Evaluating the societal relevance of academic research: A guide. *EriC-Evaluating Research in Context*, 2010. Publisher: ERiC-Evaluating Research in Context.

[3] Michael J. Carley and Eduardo Bustelo. *Social Impact Assessment And Monitoring: A Guide To The Literature*. Routledge, 2019. Google-Books-ID: lqiaDwAAQBAJ.

[4] Caroline S. Armitage, Marta Lorenz, and Susanne Mikki. Mapping scholarly publications related to the sustainable development goals: Do independent bibliometric approaches get the same results? *Quantitative Science Studies*, 1(3):1092–1108, 2020.

[5] Maurice Vanderfeesten, Eike Spielberg, and Yassin Gunes. Survey data of "mapping research output to the sustainable development goals (SDGs)". type: dataset.

[6] Maurice Vanderfeesten, Eike Spielberg, and Linda Hasse. Text analyses of survey data on "mapping research output to the sustainable development goals (SDGs)". type: dataset.

[7] Maurice Vanderfeesten, René Otten, and Eike Spielberg. Search queries for "mapping research output to the sustainable development goals (SDGs)" v5.0.2.

[8] Felix Schmidt and Maurice Vanderfeesten. Evaluation on accuracy of mapping science to the united nations' sustainable development goals (SDGs) of the aurora SDG queries.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding.

[10] Myrthe Buckens, Dynon van der Ende, and Eva den Uijl. Text mining domains: Sustainability.