Experiments with universal CEFR classification

Celia LAMRI, Hamza JEBBAR, Marwa BENAHNIA and Tinhinane BEN SLIMANE[1]

[1]Machine Leanrning and Data Science 'MLSD'

Abstract

The linguistic level is a study that has become very important in several areas, for example selecting students for university courses, knowing which language is better spoken in the world (classification). Today, we have established a replication for a study conducted by two authors Sowmya Vajjala and Taraka Rama who studied Automated Essay Scoring (AES) and described language skills on a CEFR scale, to compare their results with the results obtained after this replication and we concluded that in the three classifications monolinguistic, multilinguistic and cross-linguistic we have a remarkable difference in the field features while for the other features we had comparable results.

1. Introduction

Nowadays, NLP has a strong presence in the field of text analysis and touches on several branches. Among these is the detection of language level in various languages. In the article "Experiments with Universal CEFR Classification" by Sowmya Vajjala and Taraka Rama, the experiment is based on three languages (German (DE), Italian(IT), Czech(CZ)). The two authors sought to perform a universal classification of the Common European Framework of Reference (CEFR) using domain-specific features from the essays written by non-speaker learners in three languages, which aims at their automatic scoring (Automated essay Scoring AES) and to describe the language skills of each one on a six-level scale [A1-C2]. For this purpose, AES has been modeled by a regression, ranking and classification task.

Based on similar related studies where researchers relied on approaches that work only on one language (Monolingual: English mainly) but not on approaches that exploit cross-linguistic (training a model on one language and testing it on another) and multilingual AES (training and testing the model on several languages). Sowmya Vajjala and Taraka Rama opted for a common model that could work on all three languages.

In this project, we will establish a replication of the mentioned article in which we will reproduce the work done and then compare the results obtained with the results of the article to answer the following question: Is there a universal model for the classification of language proficiency?

2. What is a replication?

A Replication is a reproduction of an experiment under the same conditions with the same data used, metrics (evaluation methods), methods (data processing, initialization parameters, and configurations), and the same initial results (reproducible). It's important to know that a repeated experiment that does not return the same results is not scientifically significant. An experiment repeated several times returned the same results. That is not always reliable because the conditions and errors remain the same.

3. Summary of the article

The work is carried out on data extracted from the MERLIN corpus classified according to the CEFR scale for several languages and consists of 2286 written texts (formal and informal letters-mails) by non-speaker students in three secondary languages DE, IT, CZ with the aim of finding their linguistic levels of [A1-C2]. The corpus contains multidimensional linguistic annotations based on several aspects: grammatical accuracy, vocabulary range and sociolinguistic awareness.

The study was initiated by pre-processing the corpus, the removal of language-CEFR category combinations with less than 10 examples in the corpus and the removal of unrated essays from the original corpus to end up with 2266 documents.

The features used in AES systems are summarized in words and Part of speech (POS) n-grams with $n \in [1-5]$ and frequency of occurrence ≥ 10 , vectorization of words and features with softmax layers, dependency n-grame where each word is represented by the triplet (dependency relation, dependency POS tag and head POS tag), and linguistic features appropriate to the AES literature (domain features): document length, lexical richness features(of lexical density, lexical variation, lexical diversity), and error features (open source spelling/grammar errors for German and Italian but not for Czech because no resources). Note that the POS and Universal Dependency (UD) based features are extracted with a UDPipe parser. For better final classification results, they combined the n-grams feature (Sparse) and the domain features (Dense). The authors compared non-embedding feature experiments with logistic regression, random forests, multilayer perceptrons, and SVM, as well as neural network models trained on task-specific embedding representations of other experiments. To get good results from these models, they used 10-fold cross-validation for monolingual and multilingual type and language's data test for cross-lingual and implemented neural networks using Keras with tensorflow and Scikit-learn for the other models.

In order to evaluate the performance of the exploited models, they used **F1 weighted score** because the clusters are not balanced in terms of observation number.

Thus, , we present a brief discussion of the results obtained by the two authors. Due to space restrictions, the authors only stated the best performing systems in the paper and divided obtained results into 3 parts: Monolingual classification, Multilingual classification and Cross-lingual classification. The two best performing classifiers in the experiments were Random Forest and Logistic regression, as they also chose the document length as the baseline. For monolingual classification : the features used for this task are the Baseline that has the lowest results for the three languages, the Word embeddings that scored an average outcome, the three sets of n-grams that scored good results especially for the Italian language and last the n-grams plus domain features that has the best F1 scores with a significant 25% improvement compared to baseline. The use of embeddings pre-trained on a larger corpus is something that should be investigated in future. The classification results differ between different languages due to the number of classes for each language and the length of the corpus (5 classes for German and only 3 for the two other languages).

Concerning the multilingual classification, the authors used for non-neural models an extra categorical feature that represents which language the corpus is written and also provided results without using the extra feature. As in the previous classification, the baseline returned poor scores. We can easily notice that the n-grams sets return the best results for the multilingual models (POS) while Domain features still score average results. We observe that the same features score poorer than monolingual models which can be caused by more training data or the fact that some feature groups have similarities in terms of proficiency categories assigned for different languages.

Cross-lingual classification is training a CEFR model on one language and testing it on others.As mentioned before the German language is the only one that has all the categories in the corpus, the authors trained all the models on DE as they excluded word n-grams and word embeddings as they are lexical and are language specific and are not suitable for this case. Which leaves us with only two cases to compare that are testing on IT and testing on CZ.Results are dropping by 10% compared to monolingual classification for both cases which is not surprising as the feature weights are tuned to German syntactic features while this classification still captures the proficiency scale meaningfully. These results suggest possible universal patterns of language use in the progression towards language proficiency.

Overall, the results indicate that cross-lingual and multilingual classifiers returned comparable performance to individual language models which drives us to the idea of a universal notion of language proficiency.

4. Results obtained after replication/Comparison of results

Note that there are no columns for the baseline and word/word + char embeddings (for our results) because they are not included in the experiment.

4.1. Monolingual classification

The results are comparable for word ngrams, pos n-grams, and dep n-grams (with and without domain). While for domain features we obtained worse results than those obtained by the authors.

Features	**DE**	**IT**	**CZ**
Baseline	0.497	0.578^L	0.587^L
Word ngrams (1)	0.666	0.827	0.721
POS ngrams (2)	0.663	0.825	0.699
Dep.ngrams (3)	0.663	0.813	0.704
Domain features	0.533^L	0.653^L	0.663
(1) + Domain	0.686	0.837	0.734
(2) + Domain	0.686	0.816	0.709
(3) + Domain	0.682	0.806	0.712
Word embeddings	0.646	0.794	0.625

Table 1: Weighted F1 scores for Monolingual Classification (authors)

Features	**DE**	**IT**	**CZ**
Baseline	-	-	-
Word ngrams (1)	0.580	0.790	0.670
POS ngrams (2)	0.650	0.790	0.650
Dep.ngrams (3)	0.630	0.780	0.670
Domain features	0.350	0.270	0.310
(1) + Domain	0.630	0.790	0.680
(2) + Domain	0.650	0.790	0.670
(3) + Domain	0.620	0.780	0.680
Word embeddings	-	-	-

Table 2: Weighted F1 scores for Monolingual Classification (our results)

4.2. Multilingual classification

As a reminder, we label Lang (+) the result by adding the variable that contains the language of the corpus and Lang(-) the results without it.

In the code, there is no information about whether the language is processed. So we don't know what our results correspond to. There are no criteria specified in the code (word n-grams, dependent n-grams, and feature domains), so in order to compare the results, we have to add them ourselves and compare our unique results with the two provided by the author (lang(+) and lang(-)).

For Word n-grams, POS n-grams, and dep n-grams, our results are slightly worse than those obtained by the authors. Contrary to the domain feature, we had better results.

Features	**Lang(-)**	**Lang(+)**
Baseline	0.428^L	-
Word ngrams (1)	0.721	0.719
POS ngrams (2)	0.726	0.724
Dep.ngrams (3)	0.703	0.693
Domain features	0.449^L	0.471^L
Word + Char embeddings	0.693	00.689

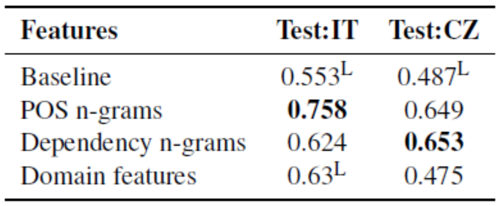
Table 3: Weighted F1 scores for multilingual classification with models trained on combined datasets (authors).

Features	**obtained results**
Baseline	-
Word ngrams	0.59
POS ngrams	0.66
Dep.ngrams	0.65
Domain features	0.57
Word + Char embeddings	-

Table 4: Weighted F1 scores for multilingual classification with models trained on combined datasets.(our results)

4.3. Cross-lingual classification

When the test data was in Italian, equivalent results were obtained for n-grams, pos n-grams, and dep n-grams, and the same results were obtained for Czech.



Features	Test:IT	Test:CZ
Baseline	0.553^L	0.487^L
POS n-grams	**0.758**	0.649
Dependency n-grams	0.624	**0.653**
Domain features	0.63^L	0.475

Table 5: Weighted F1 scores for cross-lingual classification model trained on German (authors).

Features	Test:IT	Test:CZ
Baseline	-	-
POS ngrams	0.70	0.65
Dep.ngrams	0.57	0.64
Domain features	0.36	0.47

Table 6: Weighted F1 scores for cross-lingual classification model trained on German (our results).

4.3.1. Confusion matrices

Considering the quality of the classification results given by the confusion matrices, we can see that the quality returned by DE-Train: IT-Test setup with the POS n-gram features provided by the authors are comparable. In the case of the **Table 10**, however, we can see that the quality of the result is better than **Table 9**.

→ *Pred*	**A1**	**A2**	**B1**	**B2**	**C1**
A1	5	54	0	0	0
A2	9	311	56	5	0
B1	1	70	279	44	0

Table 7: DE-Train:IT-Test setup with POS n-gram features (authors)

→ *Pred*	**A1**	**A2**	**B1**	**B2**	**C1**
A1	3	26	0	0	0
A2	9	333	30	0	0
B2	1	89	209	95	0
C1	0	0	0	0	0

Table 8: DE-Train:IT-Test setup with POS n-gram features (our results)

→ *Pred*	**A1**	**A2**	**B1**	**B2**	**C1**
A1	0	129	57	2	0
B1	0	23	101	41	0
B2	0	5	25	51	0

Table 9: DE-Train:CZ-Test setup with Dependency features (authors)

→ *Pred*	**A1**	**A2**	**B1**	**B2**	**C1**
A1	3	26	0	0	0
A2	9	333	30	0	0
B2	1	89	209	95	0
C1	0	0	0	0	0

Table 10: DE-Train:CZ-Test setup with Dependency features (our results)

4.3.2. **Suggestions**

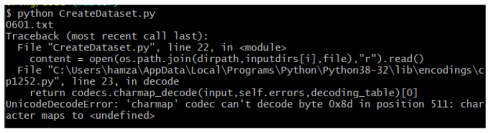
We assume that the causes for some of our results are different from those returned by the authors are :

- Due to some packages being abandoned and us using new ones, some features used in the training are no longer available, we only used the two available features while they used three,
- The features produced by the new language tool that we used although close to the language check package that they used, there are some differences that may give different results,
- We have encountered some files that have no scoring, although rare but they exist, this detail isn't mentioned in the paper, which prompts us to think that maybe there are some changes in the original files between the time they conducted their experiment and now.

5. Problems encountered

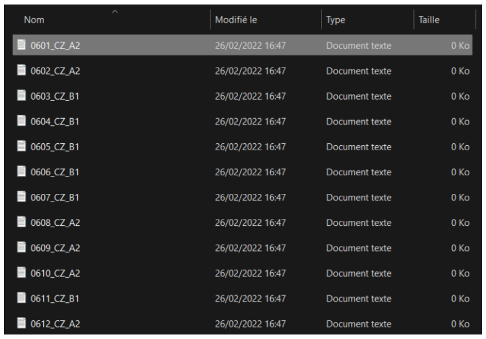
5.1. Issues

we have encountered multiple issues while executing the code:



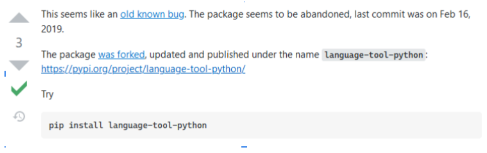
```
$ python CreateDataset.py
0601.txt
Traceback (most recent call last):
  File "CreateDataset.py", line 22, in <module>
    content = open(os.path.join(dirpath,inputdirs[i],file),"r").read()
  File "C:\Users\hamza\AppData\Local\Programs\Python\Python38-32\lib\encodings\c
p1252.py", line 23, in decode
    return codecs.charmap_decode(input,self.errors,decoding_table)[0]
UnicodeDecodeError: 'charmap' codec can't decode byte 0x8d in position 511: char
acter maps to <undefined>
```

The first one with the encoding. It seems that some files had some characters that required utf-8 encoding. We have resolved the issue by adding "encoding='utf8'" whenever we open a file.



Nom	Modifié le	Type	Taille
0601_CZ_A2	26/02/2022 16:47	Document texte	0 Ko
0602_CZ_A2	26/02/2022 16:47	Document texte	0 Ko
0603_CZ_B1	26/02/2022 16:47	Document texte	0 Ko
0604_CZ_B1	26/02/2022 16:47	Document texte	0 Ko
0605_CZ_B1	26/02/2022 16:47	Document texte	0 Ko
0606_CZ_B1	26/02/2022 16:47	Document texte	0 Ko
0607_CZ_B1	26/02/2022 16:47	Document texte	0 Ko
0608_CZ_A2	26/02/2022 16:47	Document texte	0 Ko
0609_CZ_A2	26/02/2022 16:47	Document texte	0 Ko
0610_CZ_A2	26/02/2022 16:47	Document texte	0 Ko
0611_CZ_B1	26/02/2022 16:47	Document texte	0 Ko
0612_CZ_A2	26/02/2022 16:47	Document texte	0 Ko

The second issue is that the CreateDataset.py script doesn't write its output in the files, thus producing empty files without the desired results. We resolved that by just adding a line that writes the content in the file



3

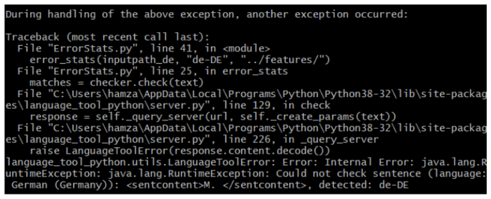
This seems like an old known bug. The package seems to be abandoned, last commit was on Feb 16, 2019.

The package was forked, updated and published under the name `language-tool-python`: https://pypi.org/project/language-tool-python/

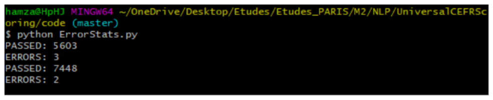
Try

```
pip install language-tool-python
```

Here we have encountered an issue with the language-check package, it had been abandoned a few years ago and the installation setup is no longer working. We replaced it with the new language-tool-python package, although they are very similar, there was some slight difference in the output that each one gave, in addition to the language-tool-python not having 'locqualityissuetype' attribute, which is one of the three features that the old implementation used, and therefore won't be used in ours.

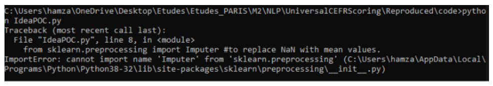


```
During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "ErrorStats.py", line 41, in <module>
    error_stats(inputpath_de, "de-DE", "../features/")
  File "ErrorStats.py", line 25, in error_stats
    matches = checker.check(text)
  File "C:\Users\hamza\AppData\Local\Programs\Python\Python38-32\lib\site-packag
es\language_tool_python\server.py", line 129, in check
    response = self._query_server(url, self._create_params(text))
  File "C:\Users\hamza\AppData\Local\Programs\Python\Python38-32\lib\site-packag
es\language_tool_python\server.py", line 226, in _query_server
    raise LanguageToolError(response.content.decode())
language_tool_python.utils.LanguageToolError: Error: Internal Error: java.lang.R
untimeException: java.lang.RuntimeException: Could not check sentence (language:
 German (Germany)): <sentcontent>M. </sentcontent>, detected: de-DE
```



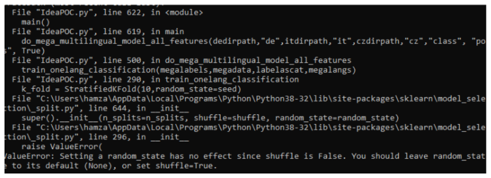
```
hamza@HpHJ MINGW64 ~/OneDrive/Desktop/Etudes/Etudes_PARIS/M2/NLP/UniversalCEFRSc
oring/code (master)
$ python ErrorStats.py
PASSED: 5603
ERRORS: 3
PASSED: 7448
ERRORS: 2
```

There were some problems with the spell-checker, we just added a try except clause to allow the script to continue, the proportion of the errors is insignificant compared to the rest of the dataset.



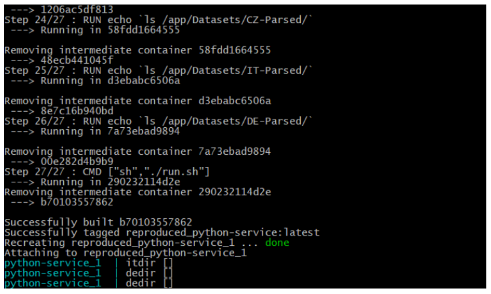
```
C:\Users\hamza\OneDrive\Desktop\Etudes\Etudes_PARIS\M2\NLP\UniversalCEFRScoring\Reproduced\code>pytho
n IdeaPOC.py
Traceback (most recent call last):
  File "IdeaPOC.py", line 8, in <module>
    from sklearn.preprocessing import Imputer #to replace NaN with mean values.
ImportError: cannot import name 'Imputer' from 'sklearn.preprocessing' (C:\Users\hamza\AppData\Local\
Programs\Python\Python38-32\lib\site-packages\sklearn\preprocessing\__init__.py)
```

This is also one of the packages that had been dropped, we replaced it with the new version "SimpleImputer".



```
  File "IdeaPOC.py", line 622, in <module>
    main()
  File "IdeaPOC.py", line 619, in main
    do_mega_multilingual_model_all_features(dedirpath,"de",itdirpath,"it",czdirpath,"cz","class", "po
s", True)
  File "IdeaPOC.py", line 500, in do_mega_multilingual_model_all_features
    train_onelang_classification(megalabels,megadata,labelascat,megalangs)
  File "IdeaPOC.py", line 290, in train_onelang_classification
    k_fold = StratifiedKFold(10,random_state=seed)
  File "C:\Users\hamza\AppData\Local\Programs\Python\Python38-32\lib\site-packages\sklearn\model_sele
ction\_split.py", line 644, in __init__
    super().__init__(n_splits=n_splits, shuffle=shuffle, random_state=random_state)
  File "C:\Users\hamza\AppData\Local\Programs\Python\Python38-32\lib\site-packages\sklearn\model_sele
ction\_split.py", line 296, in __init__
    raise ValueError(
ValueError: Setting a random_state has no effect since shuffle is False. You should leave random_stat
e to its default (None), or set shuffle=True.
```

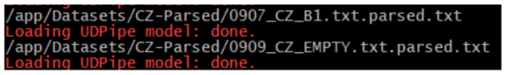
This is an error caused by a conflict of parameters' values. We resolved it by removing the "random_state" parameter.



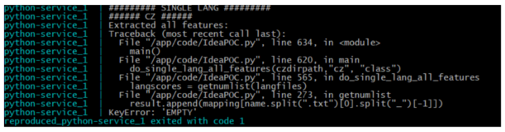
```
 ---> 1206ac5df813
Step 24/27 : RUN echo `ls /app/Datasets/CZ-Parsed/`
 ---> Running in 58fdd1664555

Removing intermediate container 58fdd1664555
 ---> 48ecb441045f
Step 25/27 : RUN echo `ls /app/Datasets/IT-Parsed/`
 ---> Running in d3ebabc6506a

Removing intermediate container d3ebabc6506a
 ---> 8e7c16b940bd
Step 26/27 : RUN echo `ls /app/Datasets/DE-Parsed/`
 ---> Running in 7a73ebad9894

Removing intermediate container 7a73ebad9894
 ---> 00e282d4b9b9
Step 27/27 : CMD ["sh","./run.sh"]
 ---> Running in 290232114d2e
Removing intermediate container 290232114d2e
 ---> b70103557862

Successfully built b70103557862
Successfully tagged reproduced_python-service:latest
Recreating reproduced_python-service_1 ... done
Attaching to reproduced_python-service_1
python-service_1  | itdir []
python-service_1  | dedir []
python-service_1  | dedir []
```

UDPIPE is supposed to parse the raw texts and write the results in new folders, with the name LANGUAGE-Parsed for each LANGUAGE in CZ, IT, DE. As you can see in this image, the UDPIPE script doesn't write anything and the folders are empty, we ran a bash command with "ls" and a python command with os.listdir juste to make sure that there is no problem with the path inside the docker, and it showed that the problem is actually in the UDPIPE script.

We resolved the problem by correcting the syntax in the bash script that calls UDPIPE, in addition to changing the UDPIPE binaries that were only compatible with Windows to Linux64 binaries for it to function in a docker container. However, the execution takes so much time to tokenize, tag and parse all the files one by one which requires loading the model each time and calling it on one file. It takes more than 2.5 hours to finish parsing all the files of the three languages.



```
/app/Datasets/CZ-Parsed/0907_CZ_B1.txt.parsed.txt
Loading UDPipe model: done.
/app/Datasets/CZ-Parsed/0909_CZ_EMPTY.txt.parsed.txt
Loading UDPipe model: done.
```



```
python-service_1  | ######### SINGLE LANG #########
python-service_1  | ###### CZ ######
python-service_1  | Extracted all features:
python-service_1  | Traceback (most recent call last):
python-service_1  |   File "/app/code/IdeaPOC.py", line 634, in <module>
python-service_1  |     main()
python-service_1  |   File "/app/code/IdeaPOC.py", line 620, in main
python-service_1  |     do_single_lang_all_features(czdirpath,"cz", "class")
python-service_1  |   File "/app/code/IdeaPOC.py", line 565, in do_single_lang_all_features
python-service_1  |     langscores = getnumlist(langfiles)
python-service_1  |   File "/app/code/IdeaPOC.py", line 273, in getnumlist
python-service_1  |     result.append(mapping[name.split(".txt")[0].split("_")[-1]])
python-service_1  | KeyError: 'EMPTY'
reproduced_python-service_1 exited with code 1
```

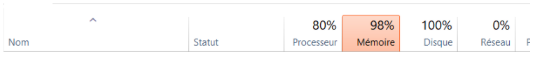
The problem we encountered here is that the script uses a python dictionary to map scores to numbers, A1 to 1 for example, and the scores are extracted from the file's name. Some files don't have a score and therefore an "EMPTY" is added instead, which raises a KeyError exception. We resolved this by adding a "try except" to the code and chose the medium B2 as a default value. We count later the number of errors to be displayed for further analysis.

Note that these observations were removed in the original experiment, but have found no code parts responsible for their removal.

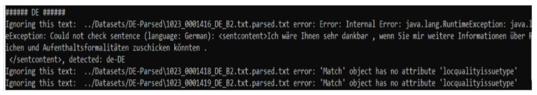
5.2. Important note

Because of the high number of code related errors that aren't even due to the packages, I believe that these scripts have not been properly tested. Therefore, recreating this experiment has been difficult, time consuming and **very unpleasant**.

5.3. Results



The final script unfortunately is very greedy, we weren't capable of running inside a docker container using our computers (12Go RAM), so we weren't capable of checking if the script does function properly or not. If you have a more powerful machine you can try to run it, but we can't guarantee if it will work or not. Next, we tried to run the scripts separately to collect and compare the results. We have encountered this issue:



```
###### DE ######
Ignoring this text:  ../Datasets/DE-Parsed\1023_0001416_DE_B2.txt.parsed.txt error: Error: Internal Error: java.lang.RuntimeException: java.l
eException: Could not check sentence (language: German): <sentcontent>Ich wäre Ihnen sehr dankbar , wenn Sie mir weitere Informationen über R
ichen und Aufenthaltsformalitäten zuschicken könnten .
 </sentcontent>, detected: de-DE
Ignoring this text:  ../Datasets/DE-Parsed\1023_0001418_DE_B2.txt.parsed.txt error: 'Match' object has no attribute 'locqualityissuetype'
Ignoring this text:  ../Datasets/DE-Parsed\1023_0001419_DE_B2.txt.parsed.txt error: 'Match' object has no attribute 'locqualityissuetype'
```

This kind of attribute is used with Italian and German but not Czech. It doesn't exist in the new package we're using (as far as we know), so we will be removing it from Italian and German as well.

6. Conclusion

Reproducing the results of this experiment was a challenge, the paper was clear and well explained, however, many of the packages that were used in the paper were either abandoned or changed. Furthermore, the coding scripts weren't properly tested to guarantee their functioning, we needed to add and change many sections to be able to reproduce all the results and metrics. The analysis of the reproduced results showed that we weren't far from what the paper ones. Finally, we understood how the different language proficiency scoring methods work. This field is open for many innovative approaches that can take text scoring to a new level.

7. Acknowledgement

For the sharing of tasks between the different members of the team, we opted for the work of

two, 2 for the development and 2 for the writing of the article, and at the end of the project, we decided to reverse the roles so that we could work on all the tasks of the project.

8. References

- https://app.grammarly.com/
- https://www.deepl.com/translator
- https://aclanthology.org/W18-0515/
- https://zenodo.org/record/1218081#.Yis1BejMJPY
- https://github.com/nishkalavallabhi/UniversalCEFRScoring