Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Estado de México

Development of advanced computer science applications (Group 301)

Results of AI tool for Plagiarism Detection

Paulo Ogando Gulias | A01751587

José Luis Madrigal Sánchez | A01745419

Cesar Emiliano Palome Luna | A01746493

Teachers:

Miguel González Mendoza

Raúl Monroy Borja

Ariel Ortíz Ramírez

Jorge Adolfo Ramirez Uresti

June 7, 2024

Index

Introduction

The project consisted of 2 phases, the first with NLP techniques and the second adding some improvement with AI.
The main code with the functions of our design is found in the PlagiarismChecker class, where the authors are mentioned at the beginning, while the functions use docstrings to give greater context.
The TextComparison file is where all the functions are called to obtain the plagiarism results of the suspicious files. It is worth mentioning that the menu is also established here to offer various analysis options (one suspect, several or all).
The PerformanceMeasure file is where true positives, true negatives, false positives, false negatives are identified to calculate the area under the curve.
All codes have their initial authoring comments, descriptive comments for certain parts of the code, and docstrings in functions. In addition, the PEP 8 python standard is respected with pycodestyle.

```
PS C:\Users\jlmad\Github\detector-plagio-equipo5> pycodestyle.exe .\PlagiarismChecker.py
PS C:\Users\jlmad\Github\detector-plagio-equipo5> pycodestyle.exe .\TextComparison.py
PS C:\Users\jlmad\Github\detector-plagio-equipo5> pycodestyle.exe .\PerformanceMeasure.py
PS C:\Users\jlmad\Github\detector-plagio-equipo5>
```

It is important to know that the code was run on a PC with 16 GB of RAM and an SSD.

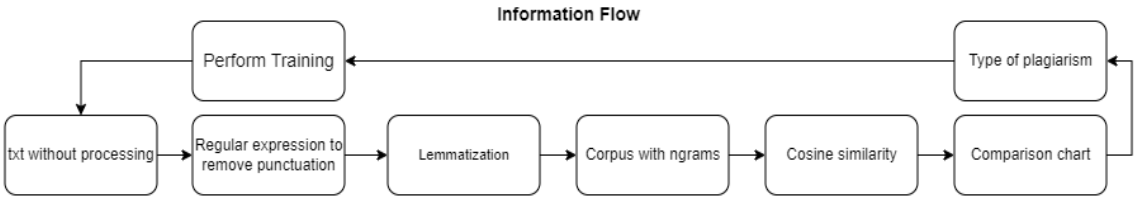
IntelligentPlagiarismChecker is the class with the functions with AI.

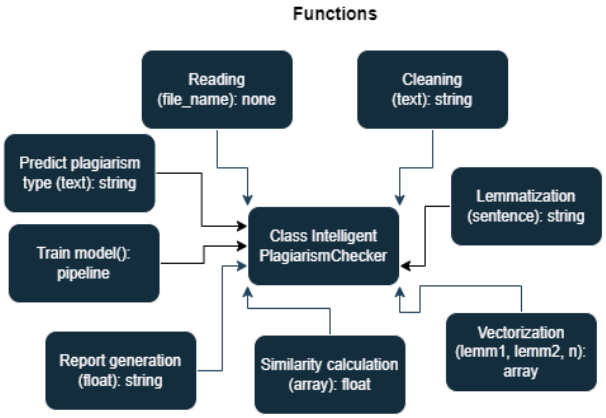
IntelligentTextComparison is where the functions are called and the different options for reviewing suspicious files are given. It takes approximately 5 minutes to run.

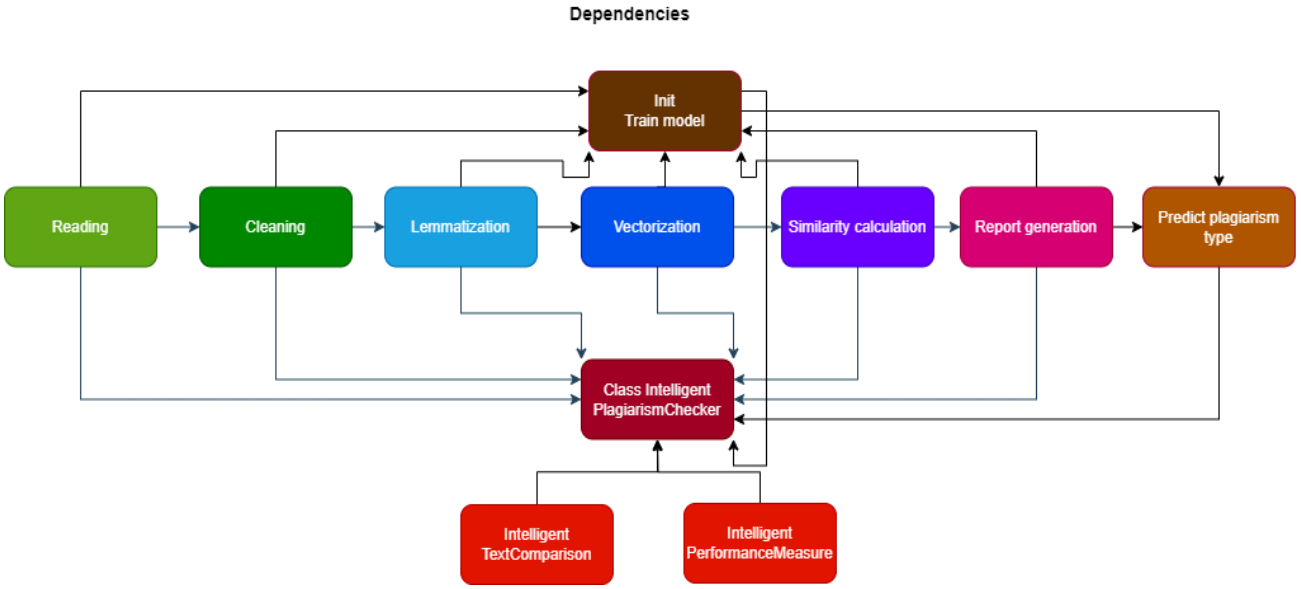
IntelligentPerformanceMeasure is where the AUC calculation is done and predictions are made with different thresholds. It takes approximately 10 minutes to run (5 minutes for training and 5 minutes for generating AUC with thresholds).

```
PS C:\Users\jlmad\Github\detector-plagio-equipo5> pycodestyle.exe .\IntelligentPlagiarismChecker.py
PS C:\Users\jlmad\Github\detector-plagio-equipo5> pycodestyle.exe .\IntelligentTextComparison.py
PS C:\Users\jlmad\Github\detector-plagio-equipo5> pycodestyle.exe .\IntelligentPerformanceMeasure.py
PS C:\Users\jlmad\Github\detector-plagio-equipo5>
```

Final design







Improvements in second phase

For the development of this project, it was decided to approach it by implementing an AI that learns through supervised learning, which allows us to train a model based on labeled data. In this way, we only implement the N-grams for plagiarism detection, and the AI model will allow us to identify the type of plagiarism present in the compared documents.

To implement this AI, we first had to conduct extensive data collection to expand our training set. Being a supervised learning model, it is extremely important to have large amounts of data, since this improves its ability to generate accurate and valuable responses, providing a frame of reference to make decisions.

Subsequently, we carried out a data labeling process so that our AI could identify the expected results, that is, the type of plagiarism present in each file. This process consists of removing special characters and accents using techniques such as stemming, as this stage is crucial to allow analysis by both unigrams and AI. This involves dividing the text into one-word sequences. These sequences provide a detailed representation of the text, crucial for detecting subtle variations that indicate plagiarism.

After this process, sklearn's random forest AI is implemented. This model allows the labeled data set to be used to train and define the characteristics of each type of plagiarism and thus be able to provide a tool that allows texts to be cataloged according to the type of plagiarism they present.

In summary, by implementing supervised learning and the Random Forest model, it is possible to develop a tool that not only indicates whether the suspected text is plagiarism, but also allows us to identify what type of plagiarism it is, thanks to the vast collection and labeling. of data.

On the other hand, it is important to mention that compared to the NLP solution, parts were added to the preprocessing of the input texts, since this helped us define a more accurate vectorization for the model. Below are the changes made:

- Definition of lowercase in cleaning: when eliminating punctuation marks, it was decided to include as an additional step that all words be in lowercase, this because greater consistency is obtained, since all words are analyzed uniformly. In fact, this can also help reduce the dimensionality of our corpus, which allows the comparison with the texts to be more efficient.
- Language identification: in the lemmatization stage, the detect method is used to know what language the text is in and thus choose the appropriate stemmer. This was done since when creating more files for training it was found that most of them were in English but there were also some in Spanish. Therefore, being able to determine the appropriate stemmer for each case is essential to obtain correct results.
- Elimination of stopwords: in both Spanish and English, there are common words that do not contribute much to the context of sentences (for example, el, la, the, is), so they are not considered in the lemmatization process, which gives us It helps make our tool even more efficient.

As mentioned, the AI model used is supervised, which consists of different elements to obtain an accuracy that generally ranges between 47 and 58%.

- Pipeline: 2 steps are established in the definition of the model, which is the conversion of texts to vectors with TfidVectorizer and a RandomForestClassifier classifier with 100 estimators is used.
- Hyper Parameters: an important part of finding the model with the conditions that offer better accuracy was the use of a grid, in which different options for various parameters are defined. GridSearchCV is used to test the different combinations, it is worth mentioning that 4 parameters were used and each 1 had between 3 or 4 options, this is because adding more takes considerable time to run the code.
- Evaluation: finally, with the grid, the best option is found based on the accuracy in order to predict the solutions with the training carried out.

It is also important to mention that from the first solution, sklearn was used to calculate the AUC with different thresholds to justify our selection. And in this installment the same functions are used to obtain the AUC both manually and with predictions with the sklearn function, which are found in the IntelligentPerformanceMeasure code.

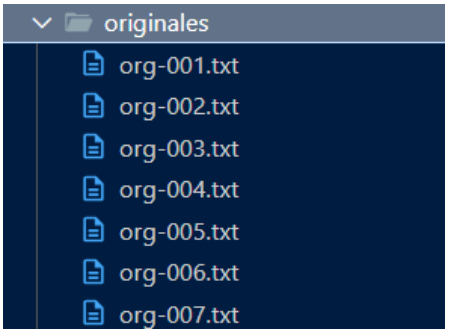
Evaluation Protocol

1. Data Set

The data set is divided into two: one for construction and the other for testing the external plagiarism detection tool. We will describe each of these below.

1.1 Construction Data Set

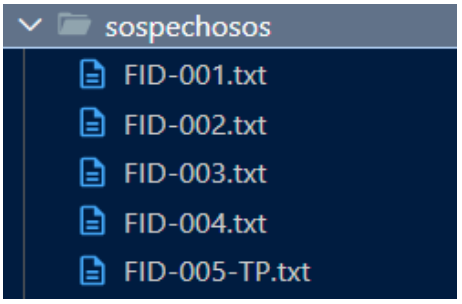
Plagiarism detection is essential to establish the originality and authenticity of a file. Our construction data set is made up of scientific articles related to different areas in artificial intelligence. In total, we have 100 abstracts, named using this format org-XX.txt (001 – 0100) and available in the “originals” folder.



1.2 Test Data Set

Our test data set consists of 111 suspicious documents, named FID-XX.txt (01 – 111). Some documents do not have plagiarism, while others were made from the original texts, this with the different methods mentioned in the next point. These are found in the “suspects” folder. It is worth mentioning that some files were named with “TP” to more easily identify the true positives, this is defined with the data given by the teachers.

Likewise, you have to know that 01 to 40 are the teachers' files, while 41 to 81 were generated on our own for training the model, while the others are to be able to evaluate the tool.



1.3 Strategies for Construction of Documents with Texts from Others

In this course, we will use one or more of the following strategies for incorporating texts from others for the purpose of masking non-originality:

A. Summary resulting from inserting or replacing phrases that belong to an original summary in another, also original.

B. Summary resulting from disordering the sentences of an original summary.

C. Summary resulting from changing tense in one or more sentences of an original summary.

D. Summary resulting from changing the voice (from active to passive or vice versa) of an original summary.

E. Summary resulting from paraphrasing an original summary, including the use of synonyms, antonyms, negation, double negation, etc.

The student should note that the strategies above are ordered by complexity in detecting non-originality. Although a tool will be evaluated using the performance measure described in the next section, we will determine the level of reach of the tool based on the maximum category of “plagiarism” that the tool can detect.

2. Performance measurement

As mentioned before, the code has a PerformanceMeasure file to calculate the area under the curve, where the true positives, true negatives, false positives and false negatives were identified to use the formulas that will give us the performance result.

$$AUC = \frac{1 + TPR - FPR}{2}$$

$$TPR = \frac{TP}{TP + FN} \text{ y } FPR = \frac{FP}{FP + TN}$$

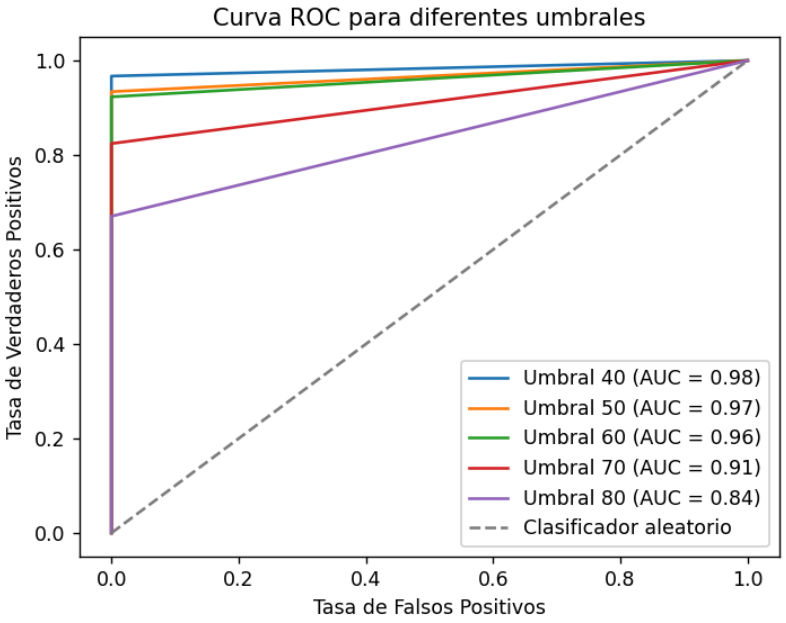
```
                    precision    recall  f1-score   support

          cambio-voz       1.00      0.25      0.40         4
desordenamiento-frases       0.00      0.00      0.00         1
   insercion-reemplazo       1.00      0.33      0.50         3
             no plagio       0.60      1.00      0.75         6
           parafraseo       0.67      0.67      0.67         3

              accuracy                           0.59        17
             macro avg       0.65      0.45      0.46        17
          weighted avg       0.74      0.59      0.56        17

Mejores parámetros: {'clf__max_depth': None, 'clf__min_samples_split': 2, 'clf__n_estimators': 100, 'tfidf__ngram_range': (1, 8)}
Precisión del modelo: 0.5882352941176471

Resultados: {'Verdaderos Positivos': 94, 'Verdaderos Negativos': 40, 'Falsos Positivos': 0, 'Falsos Negativos': 88}
AUC Calculado Manualmente: 0.7582
Umbral: 40, AUC Sklearn: 0.9835164835164836
Umbral: 50, AUC Sklearn: 0.967032967032967
Umbral: 60, AUC Sklearn: 0.9615384615384616
Umbral: 70, AUC Sklearn: 0.9120879120879121
Umbral: 80, AUC Sklearn: 0.8351648351648351
```



The best possible case of our tool came to give 58.82% accuracy (it happens approximately 1 in 3 times, but it is always certain that it will give at least 47%).
The parameters that give the best accuracy are:
'clf__max_depth': None, 'clf__min_samples_split': 2, 'clf__n_estimators': 100, 'tfidf__ngram_range': (1, 8)

Tools Comparison

Tool	Phase 2	Phase 3 with AI
TP	17	94

TN	12	40
FP	8	0
FN	3	88
Manual AUC	0.725	0.7582
Sklearn Predicted AUC	0.75	0.98
Best Sklearn Threshold	60	40

Conclusions

Firstly, it is important to mention that our tools have results with considerable differences, however, each one has a good degree of certainty taking into account the methods used and the established conditions.

For the final phase, we decided to continue determining the similarity between the original and test texts with Cosine similarity. Cosine similarity is a measure of similarity between two vectors that measures the cosine of the angle between them. In simpler terms, it is used to determine how similar two sets of data (vectors) are in a dimensional space. The differences between the previous phase and this one were eliminating capital letters and stop words in the file cleaning process. We decided to continue with said similarity since it was giving us a very favorable result in the text similarity calculation.

Our NLP solution relies directly on similarity based on word repetition, which means that it does not take into account many contextual changes or aggregations, so it simply determines whether there is plagiarism or not based on the threshold. The AUC of 0.72 indicates that the identification of plagiarism is good, although as mentioned, it is not very close to 1, this is because complex changes in the texts are not analyzed in depth. Likewise, when making predictions with the thresholds, 60 was the best case, so the condition that the similarity was greater than 59 was established to be plagiarism.

On the other hand, in our implementation with AI, the ability to identify the type of plagiarism is added, thanks to a supervised approach, where examples of each case are given to train the model, in addition to using a grid. to establish different parameters that the model can test, in order to find the best combination that has a higher accuracy, which in the best case was 58%. The AUC of 0.75 indicates that there is an improvement, although this no longer only has to do with identifying whether there is plagiarism or not, but also the type, so it is feasible that the change has not been very large, in fact, although a A dataset of 80 files for training is not small, in reality a much larger amount is needed for more accurate results. Regarding the predictions with the thresholds, in this case 40 was shown as the best, so this modification had to be made to establish whether it is plagiarism or not.

Therefore, an improvement is observed in the tool with AI, but with an even denser dataset, or even testing with more parameters, we could have an even better AUC, although it is true that this would require many more computational resources. But it is also important to analyze the threshold change that had to be made, since going from 59 to 40 indicates that our model has less tolerance for false negatives. This is because initially, in phase 2, there were several cases where percentages below 59 were not considered, which is why there were many false negatives. Therefore, defining our threshold as 40 in phase 3 allowed several of those files to already be considered plagiarism, which directly positively affects the accuracy of our model and also our area under the curve.