Relazione esercitazione – Plagiarism detection code

L'obiettivo di questa esercitazione è stato quello di sviluppare un algoritmo che fosse in grado di individuare un'eventuale plagio del codice di un programma sviluppato in Python. I files utilizzati durante la sperimentazione sono presenti nella cartella chiamata "**Risorse utili per Plagiarism detection code**". All'interno di questa cartella ci sono 4 files:

1) "File_originale.py ": file che si presume sia stato plagiato.
2) "File_plagio.py": file che è praticamente quasi identico al file originale.
3) "File_no_plagio.py": file completamente diverso da quello originale.
4) "File_intermedio.py": file che è a metà strada tra il plagio e il no plagio.

L'algoritmo sviluppato si compone dei seguenti passi:

1) Una volta presi in input i due files ("File_originale.py" e uno degli altri 3 citati pocanzi), viene eseguita l'eliminazione del carattere "\n".

2) Dopodichè, tramite l'esecuzione di opportune funzioni create ad hoc, per entrambi i files vengono estratti:

- Nomi librerie.
- Nomi classi.
- Nomi funzioni e procedure.
- Nomi argomenti delle funzioni e procedure.
- Nomi variabili.
- Commenti su singole righe e su righe multiple, per i quali vengono prima eliminate eventuali stop words e stringhe vuote e successivamente viene eseguita la lemmatizzazione. Per fare quello appena menzionato è stata utilizzata la libreria **nltk**.

3) A questo punto, vengono creati i due vettori **TF corrispondenti ai due files di input, concatenando tutte le informazioni estratte per ciascuno di essi al passo 2.** Il motivo per il quale si è deciso di utilizzare solamente la term-frequency e non la combinazione tf*idf (Term-frequency * Inverse Document Frequency) è che in questo task, ovvero quello dell'individuazione del plagio partendo da due documenti di input, probabilmente l'utilizzo dell'idf avrebbe posto a 0 il peso associato ai termini che comparivano in entrambi i file iniziali, in quanto ritenuti meno discriminativi. In questo modo però, si sarebbe persa appunto l'informazione riguardante il fatto che quel termine fosse presente effettivamente in entrambi gli script e che quindi sarebbe potuto essere un elemento importante da considerare in un eventuale tentativo di plagio.

4) Dopodichè, viene **calcolata la similarità del coseno tra i due vettori** in modo tale da capire effettivamente quanto siano simili tra loro. Durante l'esecuzione dei passi 3) e 4) è stata utilizzata la libreria **Sklearn**.

5) Infine, viene calcolata la **Longest Common Subsequence** tra i due scripts mediante l'utilizzo della libreria pylcs. Essa ci permette di poter ottenere la sottosequenza di caratteri (non necessariamente contigui) più lunga possibile che due stringhe di partenza hanno in comune. A questo punto diventa chiaro il fatto che, maggiore sarà il valore della LCS (numero intero positivo) e più sarà probabile che ci possa essere stato un plagio. Per poter ottenere comunque uno score tra 0 e 1 sfuttando la LCS, è stato deciso di eseguire questo rapporto:

$$\boldsymbol{score_{LCS}} = \frac{\mathbf{LCS}}{\boldsymbol{Numero\_caratteri\_tot\_stringa\_file\_originale}}$$

6) A questo punto i valori ottenuti dalla **similarità del coseno** e dallo $\boldsymbol{score_{LCS}}$ sono stati combinati tra loro, nel modo mostrato qui sotto, in modo da poter ottenere un valore di similarità finale (compreso sempre tra 0 e 1) tra i due files iniziali:

$$\boldsymbol{score_{Finale}} = \frac{\boldsymbol{score_{LCS}} + \boldsymbol{sim\_}\mathbf{cos}\,(\boldsymbol{v_{tf_{File_{originale}}}}, \boldsymbol{v_{tf_{File_{confronto}}}})}{\mathbf{2}}$$

I valori di soglia impostati per lo $\boldsymbol{score_{Finale}}$ sono stati i seguenti:

- Se lo score risulta essere **maggiore di 0.5**, allora l'algoritmo ci dirà che ci troviamo davanti ad un **"PLAGIO FORTE".**
- Se lo score è compreso **tra 0.30 e 0.50** (estremi inclusi), allora l'algoritmo ci dirà che siamo in presenza di un **"PLAGIO DEBOLE"**.
- Altrimenti, restituirà **"PLAGIO ASSENTE"**.

Risultati ottenuti

I test sono stati effettuati utilizzando i 4 files citati all'inizio della relazione.

Test tra: **File_originale.py - File_plagio.py**

```
sim_coseno:  0.9167961576500319
score_LCS:  0.7451708766716196
score_medio_finale:  0.8309835171608257

Plagio FORTE.
```

Test tra: **File_originale.py - File_no_plagio.py**

```
sim_coseno:  0.18988863132818753
score_LCS:  0.27414561664190196
score_medio_finale:  0.23201712398504476

Plagio ASSENTE.
```

Test tra: **File_originale.py - File_ intermedio.py**

```
sim_coseno:  0.34618769495591395
score_LCS:  0.3900445765230312
score_medio_finale:  0.36811613573947255

Plagio DEBOLE.
```