# CONFRONTO PERL E PYTHON

Nello scrivere ed eseguire i due script che sostanzialmente svolgono lo stesso compito, ho potuto appurare le affermazioni precedentemente fatte nella presente ricerca. Possiamo iniziare con confrontare il processo di installazione dei pacchetti o moduli necessari per eseguire l'analisi del sentiment sui file di testo. Per Python ho utilizzato il manager dei pacchetti PIP, per installare la libreria VADER con `pip install Vader` Una volta installato ho potuto procedere con l'esecuzione dello script senza problemi.

Per installare i moduli necessari allo script scritto in Perl ho installato il modulo Lingua::EN::Opinion che si puo installare con il comando `cpanm Lingua::EN::Opinion`. Provando a lanciare lo script dopo aver installato il modulo, viene riscontrato un errore di esecuzione, dove lo script richiede due ulteriori moduli necessari per funzionare.

```
WordNet::QueryData
WordNet::stem
```

Ho quindi provato ad installare questi due moduli come in precedenza, con il cpanm. L'installazione di entrambi i moduli NON andava a buon fine. Fortunatamente veniva creato un file di LOG con i dettagli dell'errore: nei dettagli veniva indicata la necessita di installare un ulteriore programma. Il database lessicale WordNet.

Questo database lessicale doveva essere installato e configurato e le variabili di ambiente impostate sui valori che il pacchetto si aspettava. Una volta fatto tutto cio si poteva procedere con l'installazione dei due pacchetti aggiuntivi.

## File handling

Una volta installati entrambi i moduli necessari per lo script in Perl, il passaggio necessario per andare ad implementare la sentiment analysis era andare a leggere il file di testo contenente il testo da analizzare. Questo passaggio, che risulta fondamentale e alla base di diversi compiti, che non sono necessariamente limitati al NLP, mette gia in evidenza le differenze fra i due linguaggi. In entrambi gli script ho creato due array, con le proprie sintassi

`files = [file1.txt, file2.txt, file3.txt ]`

`my @files (file1.txt, file2.txt, file3.txt);`

Si puo gia notare come perl richieda di esplicitare lo scope con la keyword `my` e il tipo di variabile [array] con la `@`, inoltre alla fine della riga troviamo il `;` che indica la chiusura della linea.

In questo caso, per il suo minimalismo, Python risulta piu leggibile. Entrambi gli script quindi provano a leggere il file. In python abbiamo un `try: with open` invece in perl è stato usato un `if else` statement, dove la condizione necessaria è `-e $filename`, questa condizione controlla l'esistenza del file (e = existence).

Perl ha diversi operatori per controllare varie caratteristiche,come permessi di scrittura, lettura etc,che ad un pubblico poco esperto possono dire molto poco, ma una volta imparati possono velocizzare di molto i vari compiti (task)

```
            with open(file_name, 'r', encoding='utf-8') as file:
                content = file.read
```

Lo script in Python utilizza la keyword `with` che si assicura che il file venga chiuso correttamente anche se si verificano errori di lettura. Alla funzione `open()` vengono poi passati tre parametri, il nome del file, la modalita di lettura `'r'` ovvero reading, e la codifica, in questo caso 'utf-8'. `file.read()` legge l'intero file e lo inserisce dentro la variabile content.

---

```
            my $content = do { local $/; <FILEHANDLE> };
```

Possiamo poi notare come in Perl sia possibile omettere le parentesi tonde, viene passato un `FH` ovvero un `file handler` e un `filename`. Se questa operazione non dovesse andare a buon fine, il programma terminerebbe con l'ultimo codice di errore espresso con la variabile `$!` .

```
my $content = do { local $/; <FILEHANDLE> };
```

Per comprendere la successiva linea di codice ho dovuto fare utilizzo di varia documentazione online.

L'espressione `do` ritorna il valore espresso fra le parentesi graffe. all' interno di questo blocco, viene dichiarata una variabile `$/` . si tratta di una variabile speciale in Perl, che corrisponde al carattere newline ovvero, `"\n"` .

In Perl se una variabile viene dichiarata ma non inizializzata è implicitamente inizializzata a `undef` . La `local` keyword viene utilizzata per creare variabili locali, in questo caso la variabile speciale sara cambiata solo all'interno di questo blocco.

Questo permettera di leggere il file all interno delle `<>` parentesi angolari in un unico passaggio. Il ritorno sara assegnato ad una variabile `$content` .

Si puo notare come ogni singole carattere in Perl sia ricco di significato, e molte opzioni possono essere omesse. Per fare un operazione cosi frequente come leggere un file, bisogna essere a conosceza di diverse keyword.