# Relazione
# Esercitazione 4

## Segmentation

## Introduzione

La seguente esercitazione ha visto l'implementazione di un algoritmo relativo al Text Tiling, per il task di segmentazione del testo, applicando step di pre-processing, usando informazioni come frequenze (globali, locali), co-occorrenze, risorse semantiche (WordNet, Nasari, etc.).

Sono stati individuati tre testi, di circa egual dimensione, riguardanti i seguenti topic

- Quantum computing
- Ebola virus
- Napoleon

NOTE:

- I testi con cui è stata svolta l'esercitazione sono stati generati interamente da ChatGPT

## Struttura del codice ed implementazione

Il codice è strutturato in tre file

- **`main.py`**
  si occupa di richiamare i principali metodi per l'esecuzione dell'algoritmo, a partire dalla parificazione del corpus iniziale fino ad arrivare al plot dei risultati

- **`nasari_utils.py`**
  si occupa di manipolare e lavorare sui vettori di Nasari, a partire dalla parificazione, dove si recupera la risorsa in formato txt e la si parsifica per un agevole accesso alle informazioni, all'implementazione della metrica di Weighted Overlap, fino ad arrivare all'implementazione del metodo di

`evaluate_similarity`, che si occupa di valutare la similarità dei segmenti intra-gruppo

- **`utils.py`**
  contiene i metodi di supporto per il pre-processing, lettura dei file, per l'individuazione dei break point, e per il plot dei risultati

Nello specifico, il task è stato realizzato eseguendo i seguenti step

- splitting
  - il testo viene letto tramite il metodo `get_corpus` e trasformato, tramite il metodo `split_corpus` in un insieme di N segmenti (paragrafi), con N numero variabile da iterare

- tokenizing
  - ogni segmento viene tokenizzato trasformandolo in un insieme di bag of words attraverso la rimozione della punteggiatura e delle stop-words.
  - la struttura dati risultate è una lista di segmenti in cui in ogni elemento sono presenti i termini rilevanti per quel segmento

- coesion ranking
  - lo scopo di questo step è quello di calcolare la coesione di un segmento rispetto ai suoi adiacenti (precedente e successivo), sulla base dei termini rilevanti presenti.
  - il valore di coesione di un segmento è dato da quanto le parole che occorrono nel segmento sono semanticamente simili alle parole che occorrono nei suoi adiacenti, di conseguenza per ogni coppia di segmenti addicenti è stata calcolata la similarità semantica per coppie di parole presenti nelle finestre.

$$Coesion(A,B) = \frac{\sum_i^{|A|} \sum_j^{|B|} sim(w_i, w_j)}{|A||B]}$$

  - è stata utilizzata come misura di similarità fra parole la Weighted Overlap basata sulla versione Lexical di NASARI.
  - per ogni segmento (corrente, precedente e successiva) vengono estratti i relativi vettori Nasari, ovvero una lista di vettori relativi a tutte le parole rilevanti presenti nel segmento. Dati i vettori v1 e

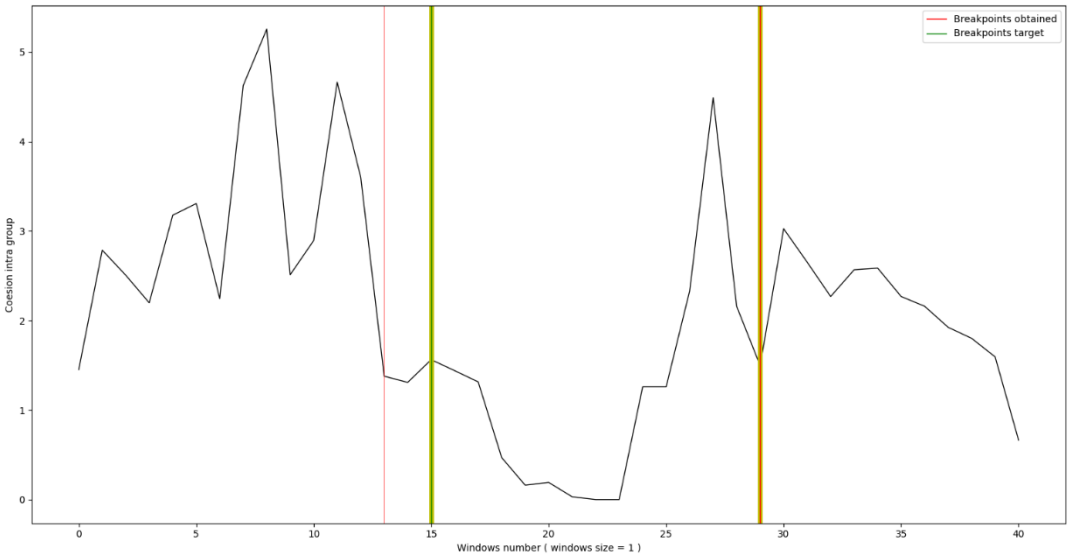
v2 viene calcolata l'intersezione tra i due e valutato il rank degli elementi comune

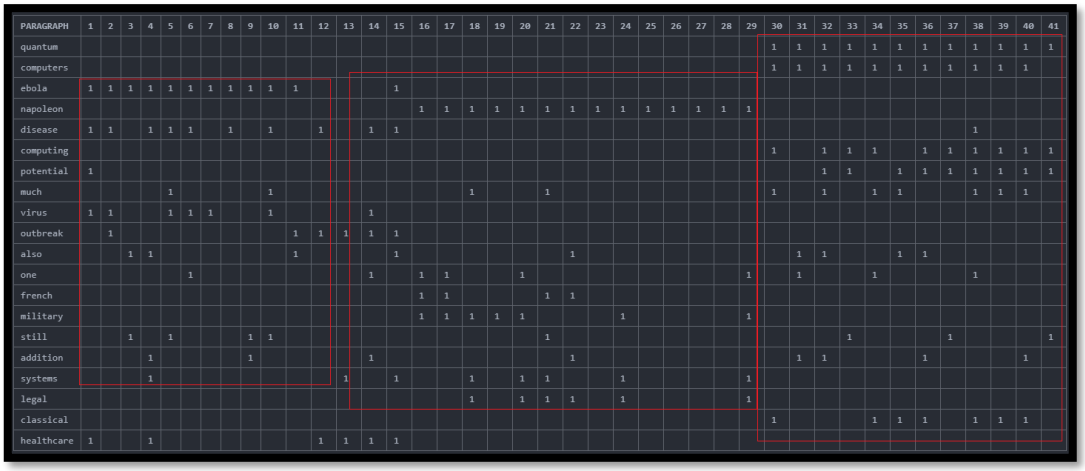
$$WO(v_1, v_2) = \frac{\sum_{q \in O} \left( rank(q, v_1) + rank(q, v_2) \right)^{-1}}{\sum_{i=1}^{|O|} (2i)^{-1}}$$

- search break point
  - riguarda la ricerca dei punti a bassa coesione circondanti da quelli ad alta coesione.
  - si cerca il miglior candidato, ovvero fra tutti i segmenti con un valore di similarità basso (sotto la media) si prende quello, la cui somma di similarità dei segmenti adiacenti è massima. L'algoritmo procede iterativamente andando ad estrarre ad ogni iterazione il miglior minimo, la condizione di stop è che non ci sono più migliori minimi da estrarre, oppure è riuscito a trovare il numero target di split points.

- plot dei risultati
  - vengono restituiti in output i risultati relativi ai break point identificati insieme ai break point corretti, e la matrice di coesione risultante

# Risultati





| PARAGRAPH | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| quantum | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| computers | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| ebola | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| napoleon | | | | | | | | | | | | | | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | |
| disease | 1 | 1 | | 1 | 1 | 1 | | 1 | | 1 | | 1 | | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | 1 | | | | |
| computing | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | | 1 | 1 | 1 | | 1 | 1 | 1 | 1 | 1 | 1 |
| potential | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 1 | | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| much | | | | | 1 | | | | | 1 | | | | | | | | 1 | | | 1 | | | | | | | | | 1 | | 1 | | 1 | 1 | | | 1 | 1 | 1 | |
| virus | 1 | 1 | | | 1 | 1 | 1 | | | 1 | | | | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| outbreak | | 1 | | | | | | | | | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| also | | | 1 | 1 | | | | | | | 1 | | | | 1 | | | | | | | 1 | | | | | | | | | 1 | 1 | | | 1 | 1 | | | | | |
| one | | | | | | 1 | | | | | | | | 1 | | 1 | 1 | | | 1 | | | | | | | | | 1 | | 1 | | | 1 | | | | 1 | | | |
| french | | | | | | | | | | | | | | | | 1 | 1 | | | | 1 | 1 | | | | | | | | | | | | | | | | | | | | |
| military | | | | | | | | | | | | | | | | 1 | 1 | 1 | 1 | 1 | | | | 1 | | | | | 1 | | | | | | | | | | | | |
| still | | | 1 | | 1 | | | | 1 | 1 | | | | | | | | | | | 1 | | | | | | | | | | | | 1 | | | | 1 | | | | 1 |
| addition | | | | 1 | | | | | 1 | | | | | 1 | | | | | | | | 1 | | | | | | | | 1 | 1 | | | | | 1 | | | | 1 | |
| systems | | | | 1 | | | | | | | | | 1 | | 1 | | | 1 | | 1 | 1 | | | 1 | | | | | 1 | | | | | | | | | | | | |
| legal | | | | | | | | | | | | | | | | | | 1 | | 1 | 1 | 1 | | 1 | | | | | 1 | | | | | | | | | | | | |
| classical | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | | | | 1 | 1 | 1 | | 1 | 1 | 1 | |
| healthcare | 1 | | | 1 | | | | | | | | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | |

## Conclusioni

Per quanto semplice come approccio, riesce ad avvicinarsi molto all'individuazione dei tagli corretti, in poche esecuzioni sul corpus. Questo per via della varietà dei termini utilizzati nei tre testi in input.

Inoltre, dai risultati si nota che la grandezza ideale di un segmento, per questo testo, è di 1 unità.

Alcuni limiti evidenziati

- scala male su testi grossi: purtroppo l'algoritmo per come è stato studiato scala male su corpus più grossi perché deve esaminare più righe di testo e deve provare più combinazioni di tagli, essendoci più righe ed eventualmente più tagli. Questo è dovuto anche al carico di risorse per la gestione dei vettori di Nasari.

- corpus: l'algoritmo performa bene su questo corpus perché in ogni segmento è presente una grande quantità di termini "*unici*", o meglio che compaiono solamente all'interno di quel segmento. Se si è tentato di applicare questo algoritmo ad un corpus con documenti più simili, ottenendo risultati più imprecisi.