Università degli studi di Torino
Dipartimento di informatica

Progetto di Tecnologie del Linguaggio Naturale

Implementazione di un sistema di text segmentation

Autori:
Christian Quaranta
Giulia Coucourde
Leonardo Magliolo

Anno Accademico 2022/2023

Introduzione

Il progetto consiste nella realizzazione di un sistema di text segmentation ispirato al funzionamento di Text-Tiling: una tecnica non supervisionata e iterativa in grado di segmentare un documento in paragrafi relazionati a sottoargomenti.

In particolare, la procedura precedentemente citata si articola in cinque fasi eseguite sequenzialmente:

1. Pre-elaborazione del testo: Inizialmente, il testo viene suddiviso in paragrafi e vengono rimossi i caratteri di punteggiatura e le parole comuni (stopword) per ridurre il rumore e focalizzarsi sugli aspetti semantici.

2. Calcolo della similarità delle frasi: Viene calcolata la similarità tra le frasi adiacenti nel testo. Questo calcolo può essere basato su misure di similarità, come la similarità del coseno o la distanza euclidea tra i vettori delle parole.

3. Creazione di profili di coerenza: L'algoritmo individua le discontinuità nella similarità tra le frasi e utilizza queste informazioni per creare i "profili di coerenza" del testo. I profili di coerenza mostrano i cambiamenti nella struttura tematica del testo, evidenziando le sezioni in cui cambiano gli argomenti trattati.

4. Segmentazione del testo: I punti di discontinuità rilevati nei profili di coerenza vengono utilizzati per dividere il testo in segmenti chiamati "tiles". Ogni tile rappresenta una sezione del testo che tratta un argomento specifico o una tematica coerente.

5. Raffinamento iterativo: Dato che la segmentazione iniziale potrebbe non essere perfetta, l'algoritmo Text-Tiling esegue più passaggi iterativi per migliorare la precisione della segmentazione. Durante ogni iterazione, il testo viene suddiviso nuovamente, e i profili di coerenza vengono aggiornati in base alle nuove informazioni ottenute.

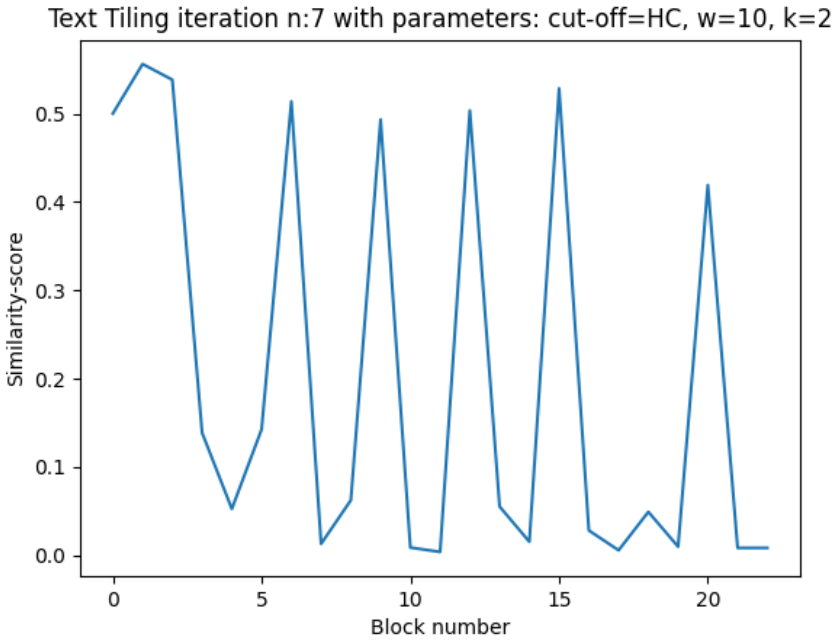
Progettazione

Facendo riferimento al paper [1], è stato scelto in fase di progettazione di adottare le seguenti specifiche circa le fasi descritte nel paragrafo precedente:

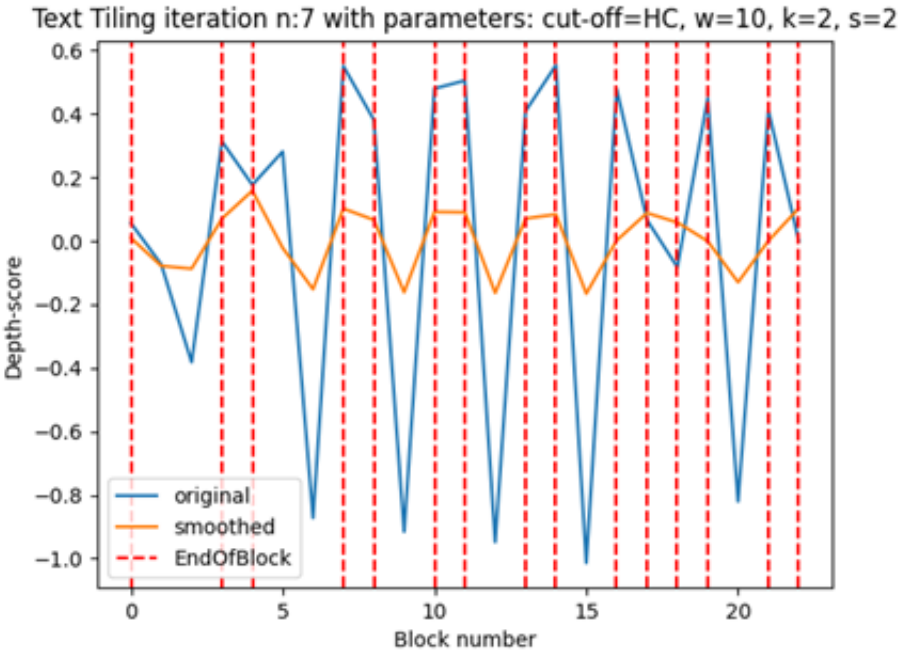
- In fase di pre-elaborazione, in aggiunta alla tokenization e alla rimozione delle stopwords, è stato deciso di effettuare operazioni di stemming per uniformare il più possibile il vocabolario estratto dal documento.
- Il documento viene diviso in pseudo-frasi di pari dimensione, definita all'interno del paper come la costante w.
  Alla prima iterazione dell'algoritmo vengono determinati blocchi di pseudo-frasi contigui e di pari dimensione, definita all'interno del paper come la costante k.
- Per valutare la similarità tra blocchi è stato scelto di rappresentare gli stessi per mezzo del formalismo Vector Space Model: ad ogni pseudo-frase viene associato un vettore 'bag-of-words' in relazione al dizionario estratto nella fase precedente, ad ogni blocco viene associato un vettore pari alla somma dei vettori associati alle pseudo-frasi appartenenti al blocco stesso.
  Ogni coppia di blocchi adiacenti verrà comparata per mezzo della metrica cosine similarity.
- Per effettuare il sub-task di boundary identification [2] è stato particolarmente utile assegnare un depth-score per ogni valore di similarità associato ad ogni coppia di blocchi adiacenti.
  Lo score è stato calcolato mediante la seguente formula:
  $$score(a_i) = \left(y_{a(i-1)} - y_{ai}\right) + \left(y_{a(i+1)} - y_{ai}\right)$$
  Per ogni $i \in (1,\ |B|)$, con B insieme dei blocchi considerati.

  Inoltre, è stata implementata una funzione di smoothing [3] in grado di appianare picchi dalla ridotta dimensione in relazione ad un intorno di valori prestabilito, indicato come costante s nel paper.
- Per calcolare una soglia che determini il numero di boundaries è possibile utilizzare due metriche indicate nel paper [3]:
  - Liberal measure (LC): $\overline{s} - \sigma$
  - Conservative measure (HC): $\overline{s} - \frac{\sigma}{2}$, per un miglior trade-off tra precision e recall rispetto a LC.

  Data una sequenza di depth-scores, calcolati $\bar{s}$ e σ su di essa, sarà possibile scartare tutte le boundaries che producano uno score minore della metrica scelta.



Esempio di valori di cosine-similarity calcolati tra blocchi adiacenti.



Esempio di depth-scores calcolati prima e dopo le operazioni di smoothing, con annessi blocchi elaborati.

Implementazione

In aggiunta alle librerie native di Python, sono state utilizzate:

- Numpy e Pandas: Per rappresentare sotto forma vettoriale e matriciale i dati necessari alla computazione delle boundaries.
  L'utilizzo di Numpy e Pandas ha permesso di ridurre i tempi di computazione delle soluzioni e di uniformare l'implementazione ad uno dei più comuni standard di programmazione per sistemi di AI in Python.
- NLTK: Utilizzata in fase di pre-elaborazione per effettuare le operazioni di tokenization, rimozione delle stopwords e stemming.
- Scikit: Utilizzata per l'implementazione fornita della cosine similarity.
- Matplotlib: Utilizzata per produrre automaticamente grafici utili al debug e all'analisi dei dati.

Test delle prestazioni

Per la composizione del documento di test sono state utilizzate tre pagine di Wikipedia relazionate a tematiche tra loro differenti (Fotografia, Cristianesimo e Astronomia).
Il testo ottenuto risulta esser composto di circa 25.000 parole.

Considerato il ridotto tempo di computazione per un documento di medio-piccole dimensioni (~ 50 secondi nel caso peggiore per il documento di test scelto) su macchina aventi le seguenti specifiche hardware:

- CPU: AMD Ryzen 7 5800X
- RAM: 32 GByte, DDR4, 3600 MHz

E' stato possibile computare accuracy, recall e F1-score del modello variando la grandezza del blocco (k) da 1 a 20 e variando il numero di iterazioni (n) da 1 a 20, fissando il numero di token della generica pseudo-frase a 10.
Ciò è stato fatto per entrambe le metriche utilizzate per determinare la soglia dei blocchi: LC e HC.

Per ciascuna delle metriche scelte è stata individuata la coppia (k, n) che è stata in grado di massimizzare l'F1-score sul documento utilizzato per il test:

- Con metrica LC: (K = 7, W = 10, N = 7, S = 2)
  - Precision: 0.95
  - Recall: 0.70
  - F1-score: 0.809
- Con metrica HC: (K = 2, W = 10, N = 10, S = 2)
  - Precision: 0.84
  - Recall: 0.828
  - F1-score: 0.834

La metrica utilizzata per calcolare precision e recall del modello sfrutta coppie di pseudo-frasi ordinate (PF1, PF2):

- Se PF1 e PF2 appartengono allo stesso blocco (nella partizione indicata come corretta) e il modello le accoppia nello stesso blocco, allora (PF1, PF2) verrà considerata un True Positive
- Se PF1 e PF2 appartengono allo stesso blocco (nella partizione indicata come corretta) e il modello non le accoppia nello stesso blocco, allora (PF1, PF2) verrà considerata un False Negative
- Se PF1 e PF2 non appartengono allo stesso blocco (nella partizione indicata come corretta) e il modello le accoppia nello stesso blocco, allora (PF1, PF2) verrà considerata un False Positive
- Se PF1 e PF2 non appartengono allo stesso blocco (nella partizione indicata come corretta) e il modello non le accoppia nello stesso blocco, allora (PF1, PF2) verrà considerata un True Negative

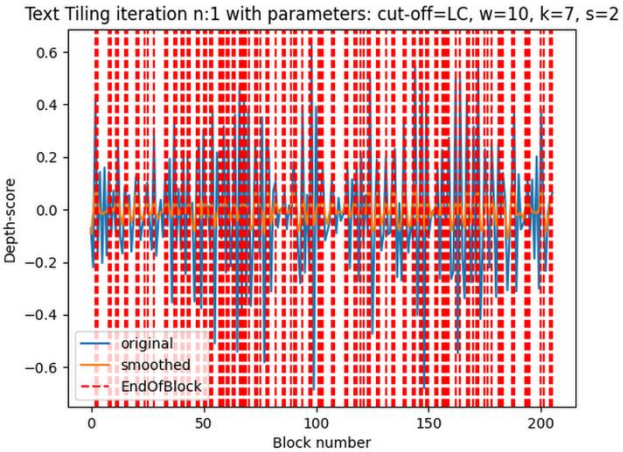
Per individuare la corretta partizione del documento sono stati inseriti all'interno del testo dei caratteri speciali usati dall'algoritmo per delineare le giuste linee di taglio.

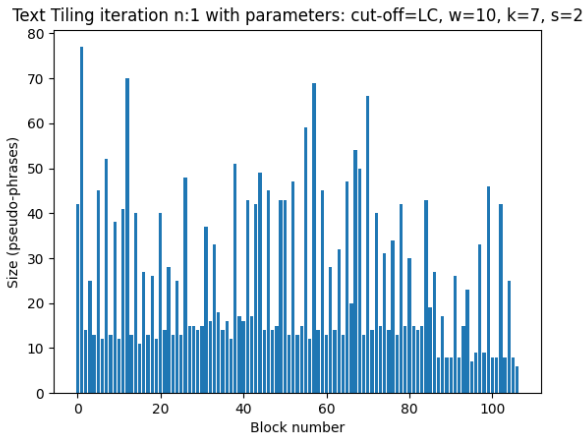
Vengono riportati di seguito coppie di grafici che identificano:

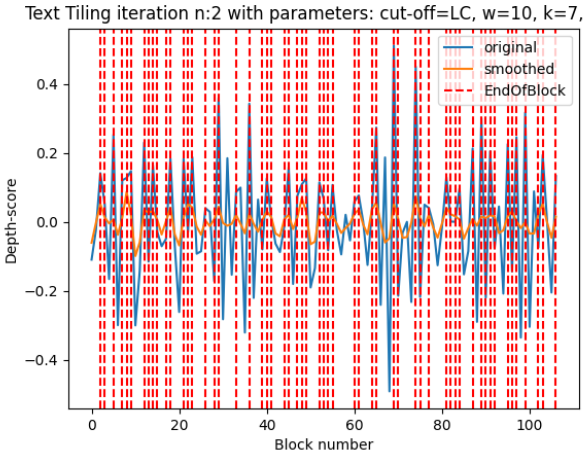
- Lunghezza di ciascun blocco alla fine di ogni iterazione (in pseudo-frasi)
- Depth-scores e linee di taglio calcolate alla fine dell'iterazione

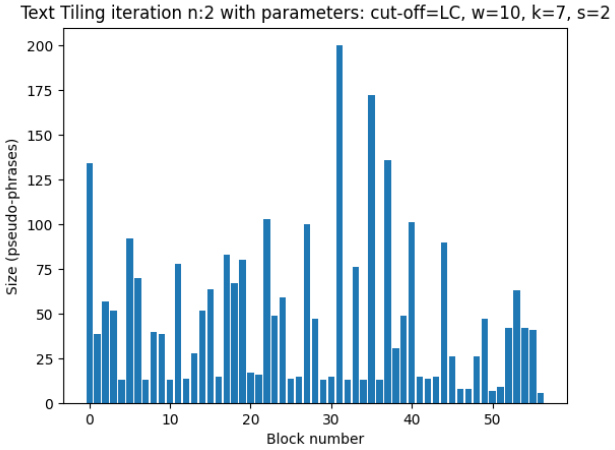
I valori sono stati calcolati per la soluzione indicata precedentemente con soglia LC:

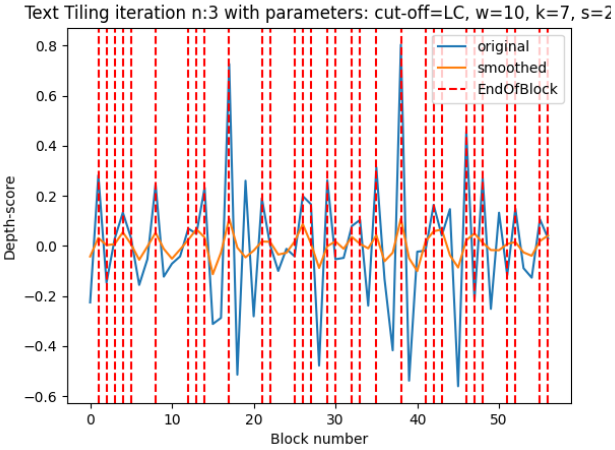
(K = 7, W = 10, N = 7, S = 2)

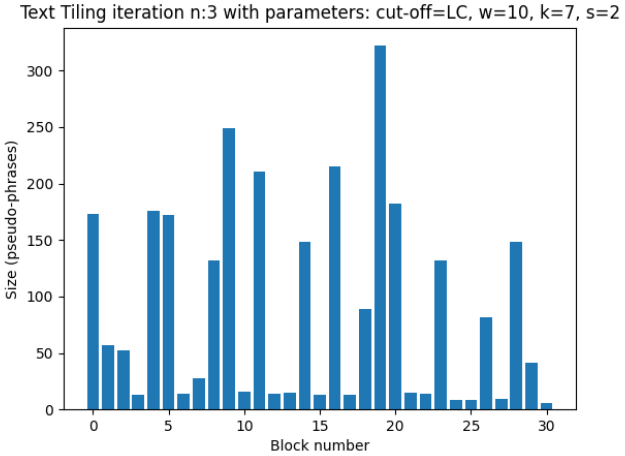


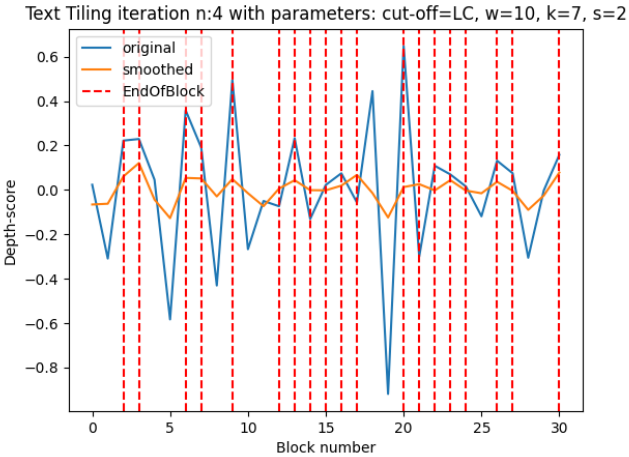


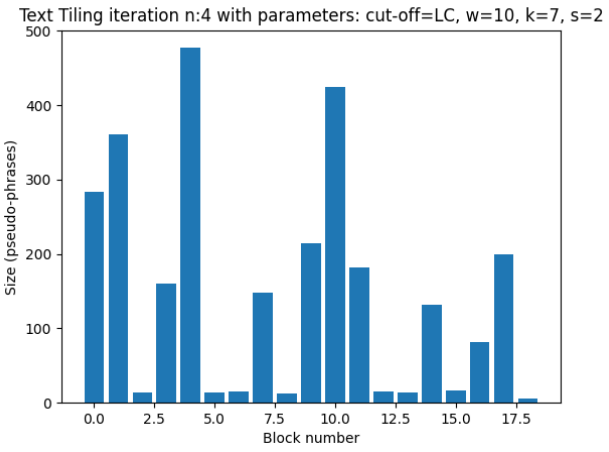

Text Tiling iteration n:2 with parameters: cut-off=LC, w=10, k=7, s=2
original
smoothed
EndOfBlock
Depth-score
Block number

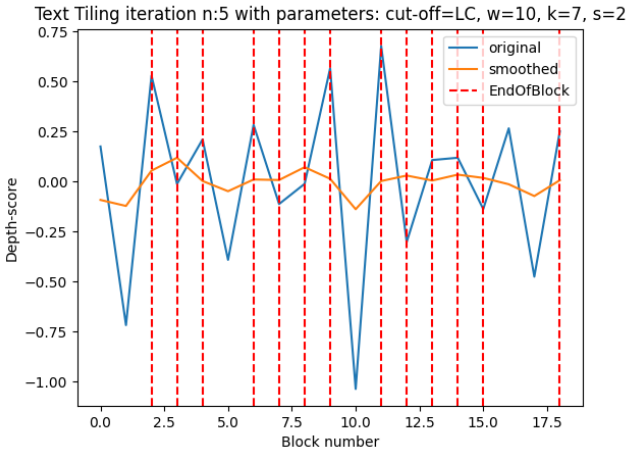

Text Tiling iteration n:2 with parameters: cut-off=LC, w=10, k=7, s=2
Size (pseudo-phrases)
Block number

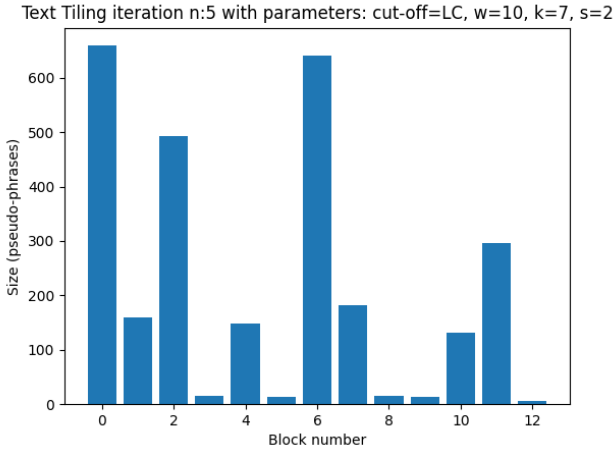

Text Tiling iteration n:3 with parameters: cut-off=LC, w=10, k=7, s=2
original
smoothed
EndOfBlock
Depth-score
Block number

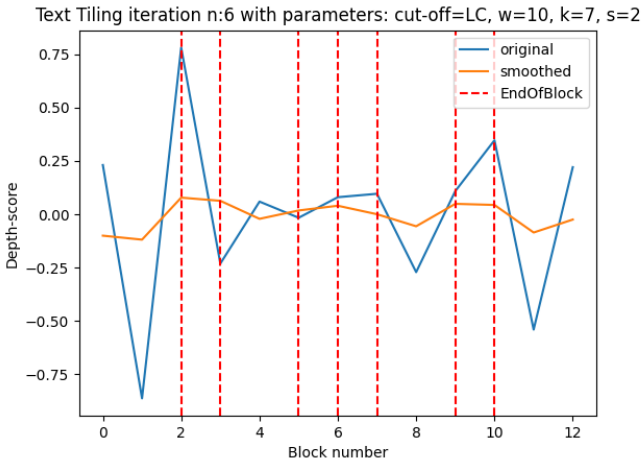

Text Tiling iteration n:3 with parameters: cut-off=LC, w=10, k=7, s=2
Size (pseudo-phrases)
Block number

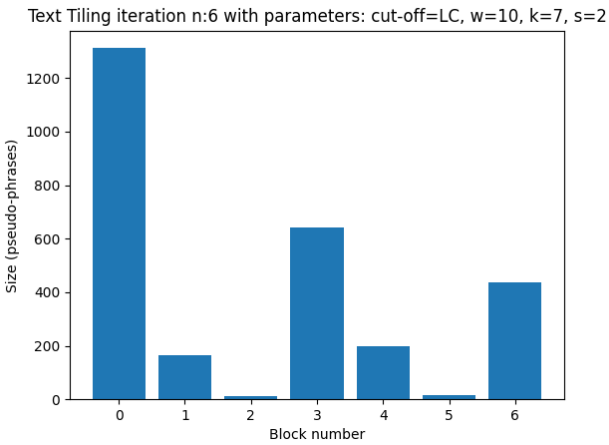

Text Tiling iteration n:4 with parameters: cut-off=LC, w=10, k=7, s=2
original
smoothed
EndOfBlock
Depth-score
Block number

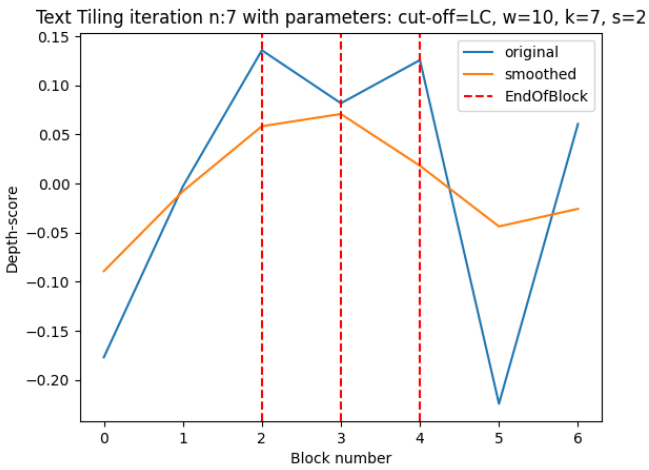

Text Tiling iteration n:4 with parameters: cut-off=LC, w=10, k=7, s=2
Size (pseudo-phrases)
Block number

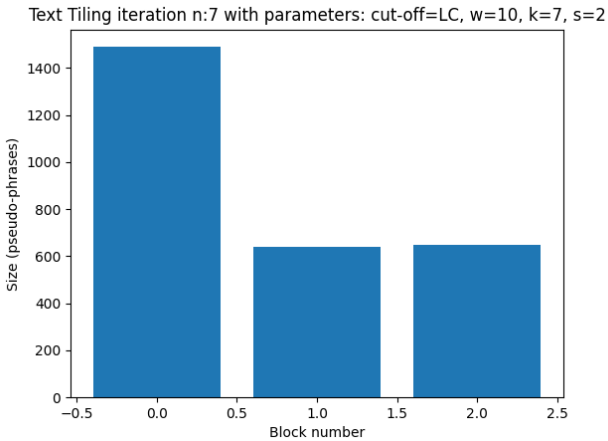

Text Tiling iteration n:5 with parameters: cut-off=LC, w=10, k=7, s=2
original
smoothed
EndOfBlock
Depth-score
Block number


Text Tiling iteration n:5 with parameters: cut-off=LC, w=10, k=7, s=2
Size (pseudo-phrases)
Block number


Text Tiling iteration n:6 with parameters: cut-off=LC, w=10, k=7, s=2
original
smoothed
EndOfBlock
Depth-score
Block number


Text Tiling iteration n:6 with parameters: cut-off=LC, w=10, k=7, s=2
Size (pseudo-phrases)
Block number


Text Tiling iteration n:7 with parameters: cut-off=LC, w=10, k=7, s=2
original
smoothed
EndOfBlock
Depth-score
Block number


Text Tiling iteration n:7 with parameters: cut-off=LC, w=10, k=7, s=2
Size (pseudo-phrases)
Block number

Analisi dei risultati

E' stato possibile riscontrare come i valori di precision e recall ottenuti per le metriche LC e HC (nei casi in cui l'F1-score sia risultato il massimo) siano comparabili a quelli riportati nel paper citato per il documento "Stargazer" suddiviso in 21 paragrafi. [4]

E' stato osservato, inoltre, come l'utilizzo della metrica HC abbia prodotto valori più alti di recall a discapito della precision, così come indicato all'interno del paper.

Fonti

1. Marti A. Hearst. Text Tiling: Segmenting Text into Multi-paragraph Subtopic Passages. Computational Linguistics, Volume 23, Number 1, March 1997; MIT Press.
2. Marti A. Hearst. Text Tiling: Segmenting Text into Multi-paragraph Subtopic Passages. "5.3 Boundary Identification", Computational Linguistics, Volume 23, Number 1, March 1997; MIT Press. p. 50.
3. Marti A. Hearst. Text Tiling: Segmenting Text into Multi-paragraph Subtopic Passages. "5.5 Determining the Number of Boundaries", Computational Linguistics, Volume 23, Number 1, March 1997; MIT Press. p. 52.
4. Marti A. Hearst. Text Tiling: Segmenting Text into Multi-paragraph Subtopic Passages. "6.3 Results: Qualitative Analysis", Computational Linguistics, Volume 23, Number 1, March 1997; MIT Press. p. 56.