University of Turin

Department of Computer Science

Natural Language Technologies Project

Implementation a text segmentation system

Authors:
Christian Quaranta
Giulia Coucourde
Leonardo Magliolo

Academic Year 2022/2023

Introduction

The project consists of the implementation of a text segmentation system inspired by the operation of Text-Tiling: an unsupervised, iterative technique capable of segmenting a document into paragraphs related to subtopics.

Specifically, the previously mentioned procedure consists of five steps performed sequentially:

1. Text pre-processing: Initially, the text is divided into paragraphs and punctuation characters and common words (stopwords) are removed to reduce noise and focus on semantic aspects.

2. Sentence similarity calculation: The similarity between adjacent sentences in the text is calculated. This calculation can be based on similarity measures such as cosine or Euclidean distance between word vectors.

3. Creation of coherence profiles: The algorithm identifies discontinuities in the similarity sentences and uses this information to create "coherence profiles" of the text. The coherence profiles show changes in the thematic structure of the text, highlighting sections where the topics covered change.

4. Text segmentation: The points of discontinuity detected in the coherence profiles are used to divide the text into segments called "tiles." Each tile represents a section of the text that deals with a specific topic or coherent theme.

5. Iterative Refinement: Since the initial segmentation may not be perfect, the Text-Tiling algorithm performs multiple iterative steps to improve the accuracy of the segmentation. During each iteration, the text is re-segmented, and the consistency profiles are updated based on the newly obtained information.

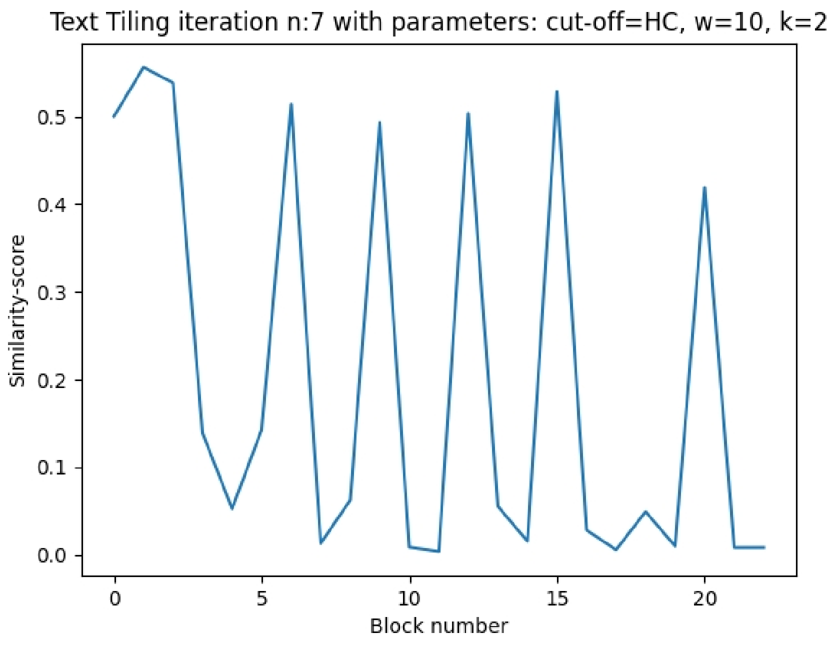
Design

Referring to the paper [1], it was chosen at the design stage to adopt the following specifications about the steps described in the previous paragraph:

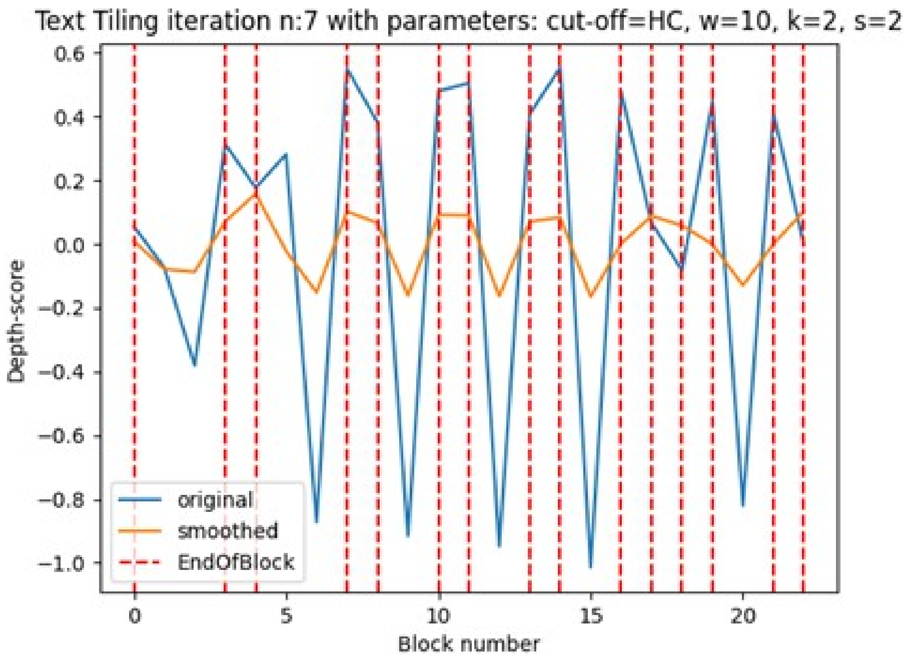
- At the pre-processing stage, in addition to tokenization and removal of stopwords, it was decided to carry out stemming operations to make the vocabulary extracted from the document as uniform as possible.
- The document is divided into pseudo-phrases of equal size, defined within the paper as the constant w.
  At the first iteration of the algorithm, contiguous pseudo-phrase blocks of equal size, defined within the paper as the constant k, are determined.
- In order to assess the similarity between blocks, it was chosen to represent them by means of the Vector Space Model formalism: each pseudo-phrase is associated with a 'bag-of-words' vector in relation to the dictionary extracted in the previous step; each block is associated with a vector equal to the sum of the vectors associated with the pseudo-phrases belonging to that block.
  Each pair of adjacent blocks will be compared by means of the cosine similarity metric.
- To perform the boundary identification sub-task [2], it was particularly useful to assign a depth-score for each similarity value associated with each pair of adjacent blocks.
  The score was calculated by the following formula:
  $$score(a_i) = (y_{a(i-1)} - y_{a_i}) + (y_{a(i+1)} - y_{a_i})$$
  For each $i \in (1, |B|)$, with B set of blocks considered.

  In addition, a smoothing function [3] has been implemented that can smooth out peaks with a small size relative to a predetermined range of values, denoted as constant s in the paper.
- Two metrics outlined in the paper [3] can be used to calculate a threshold to determine the number of boundaries:
  - Liberal measure (LC): $\bar{s} - \sigma$
  - Conservative measure (HC): $\bar{s} - \frac{\sigma}{2}$, for a better trade-off between precision and recall compared with LC.

  Given a sequence of depth-scores, calculated $\bar{s}$ and σ on it, it will be possible to discard All boundaries that produce a score lower than the chosen metric.



Example of cosine-similarity values calculated between adjacent blocks.



Example of depth-scores calculated before and after smoothing operations, with attached processed blocks.

Implementation

In addition to the native Python libraries, the following were used:

- Numpy and Pandas: To represent in vector and matrix form the data needed for boundary computation.
  The use of Numpy and Pandas reduced the computation time of the solutions and standardized the implementation to one of the most common programming standards for AI systems in Python.
- NLTK: Used in preprocessing to perform tokenization, stopword removal and stemming operations.
- Scikit: Used for provided implementation of cosine similarity.
- Matplotlib: Used to automatically produce graphs useful for debugging and data analysis.

Performance testing

Three Wikipedia pages related to different topics among them (Photography, Christianity and Astronomy) were used to compose the test document.
The resulting text turns out to be about 25,000 words.

Given the reduced computation time for a small to medium-sized document (~50 seconds in the worst case for the chosen test document) on machine having the following hardware specifications:

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

It was possible to compute accuracy, recall and F1-score of the model by varying the block size (k) from 1 to 20 and varying the number of iterations (n) from 1 to 20, fixing the number of tokens of the generic pseudo-phrase at 10.
This was done for both metrics used to determine the block threshold: LC and HC.

For each of the chosen metrics, the pair (k, n) that was able to maximize the F1-score on the document used for testing was identified:

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

The metric used to calculate precision and recall of the model exploits ordered pseudo-phrase pairs (PF1, PF2):

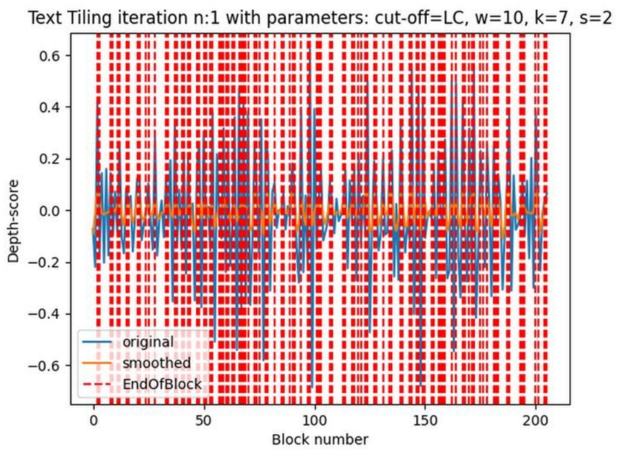
- If PF1 and PF2 belong to the same block (in the partition shown as correct) and the model pairs them in the same block, then (PF1, PF2) will be considered a True Positive
- If PF1 and PF2 belong to the same block (in the partition indicated as correct) and the model does not pair them in the same block, then (PF1, PF2) will be considered a False Negative
- If PF1 and PF2 do not belong to the same block (in the partition shown as correct) and the model pairs them in the same block, then (PF1, PF2) will be considered a False Positive
- If PF1 and PF2 do not belong to the same block (in the partition shown as correct) and the model does not match them in the same block, then (PF1, PF2) will be considered a True Negative

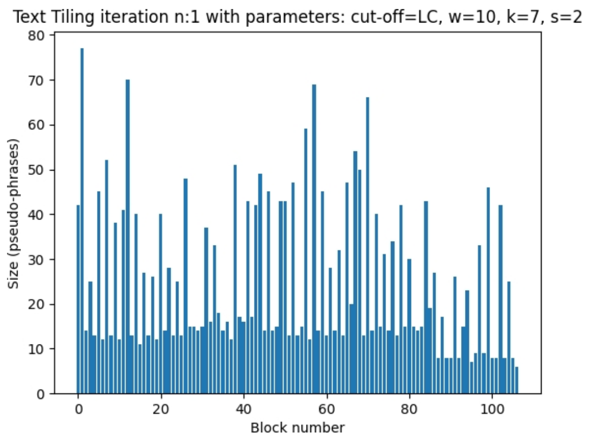
To identify the correct partition of the documentspecial characters used by the algorithm to delineate the correct cut lines were inserted within the text.

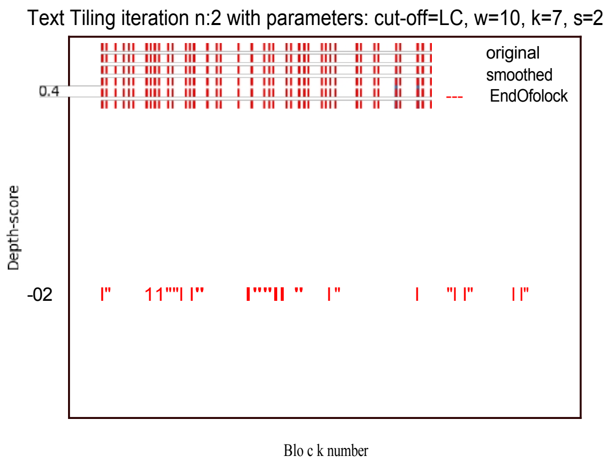
Pairs of graphs identifying are given below:

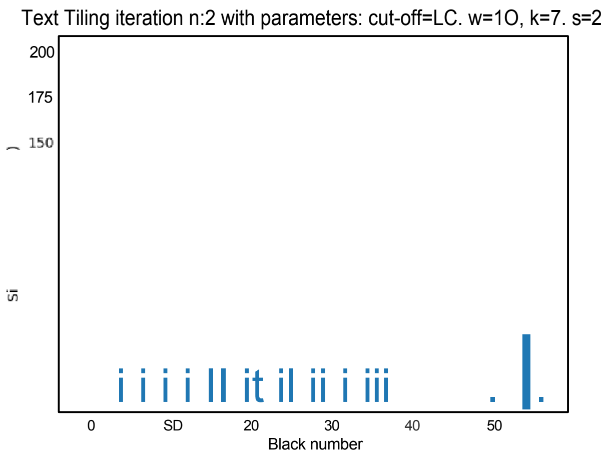
- Length of each block at the end of each iteration (in pseudo-phrases)
- Depth-scores and shear lines calculated at the end of the iteration

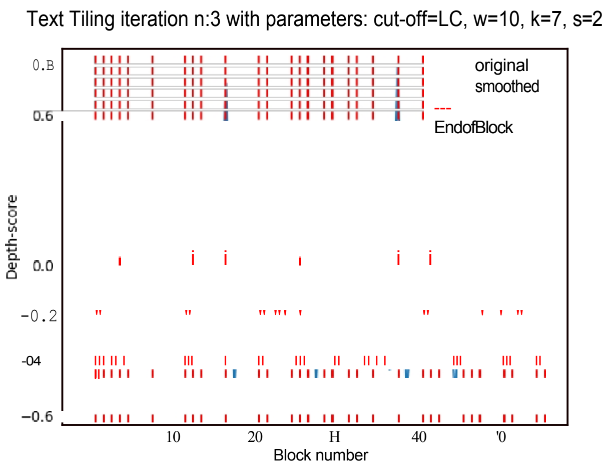
Values were calculated for the solution shown above with LC threshold: (K = 7, W = 10, N = 7, S = 2)

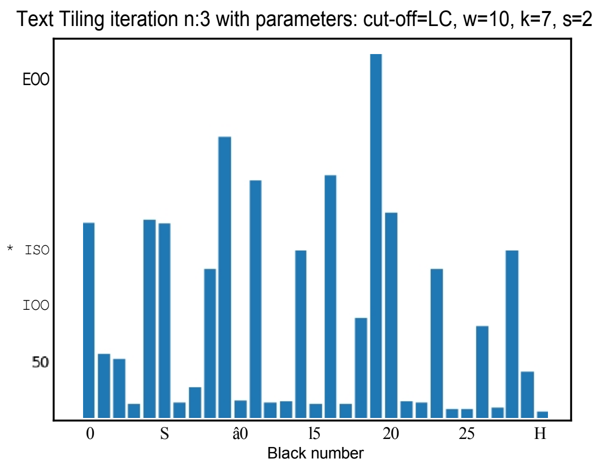


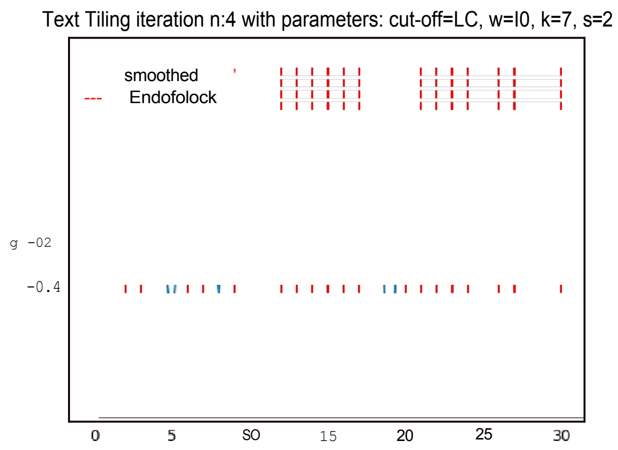


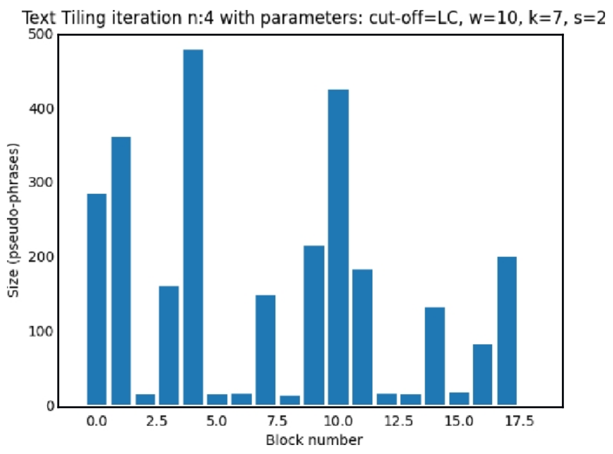

Text Tiling iteration n:2 with parameters: cut-off=LC, w=10, k=7, s=2
original
smoothed
EndOfolock
Depth-score
0.4
-02
Blo c k number

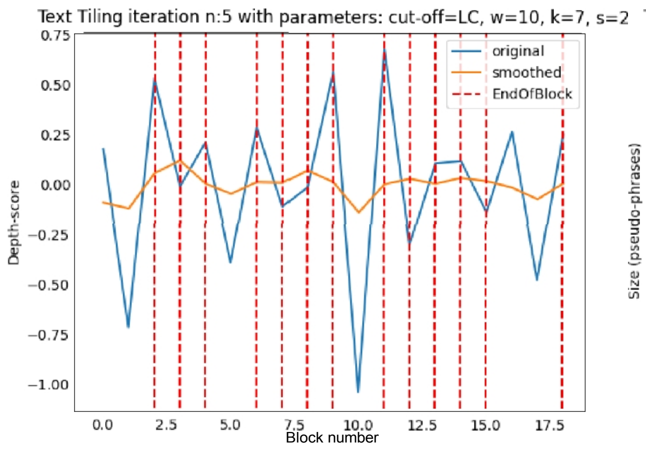

Text Tiling iteration n:2 with parameters: cut-off=LC. w=1O, k=7. s=2
200
175
150
0
SD
20
30
40
50
Black number

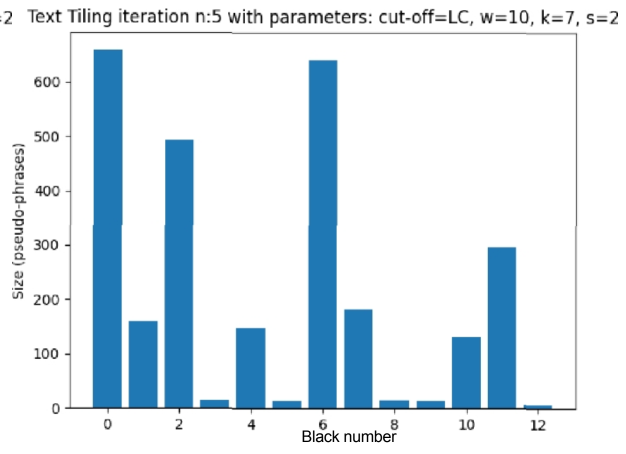

Text Tiling iteration n:3 with parameters: cut-off=LC, w=10, k=7, s=2
original
smoothed
EndofBlock
Depth-score
0.B
0.6
0.0
-0.2
-04
–0.6
10
20
H
40
'0
Block number

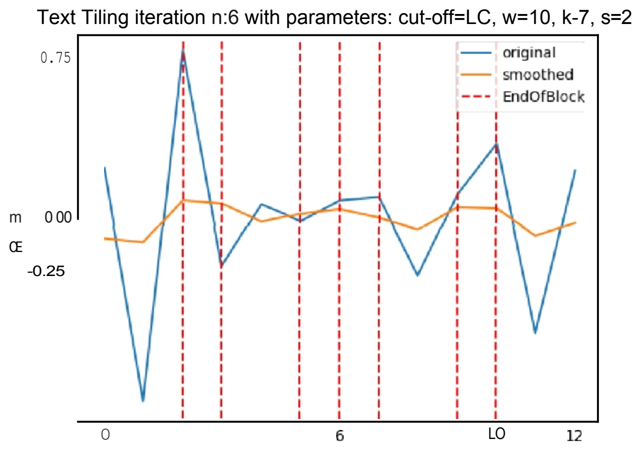

Text Tiling iteration n:3 with parameters: cut-off=LC, w=10, k=7, s=2
E00
ISO
IOO
50
0
S
â0
I5
20
25
H
Black number

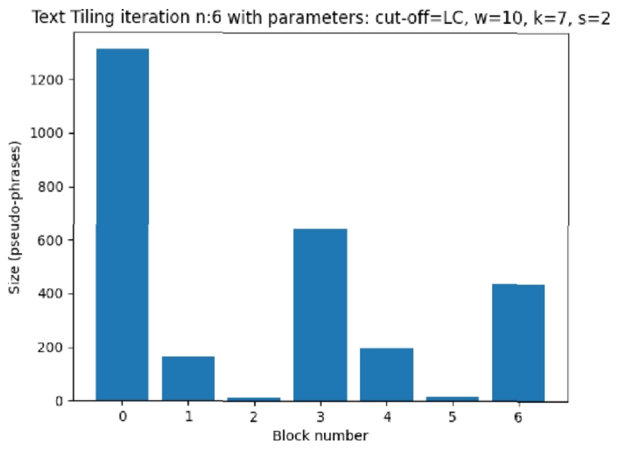

Text Tiling iteration n:4 with parameters: cut-off=LC, w=I0, k=7, s=2
smoothed
Endofolock
-02
-0.4
0
5
SO
15
20
25
30

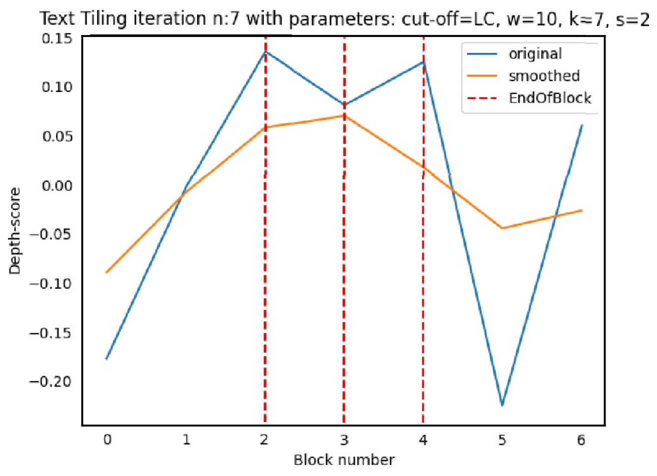

Text Tiling iteration n:4 with parameters: cut-off=LC, w=10, k=7, s=2
Size (pseudo-phrases)
500
400
300
200
100
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Block number

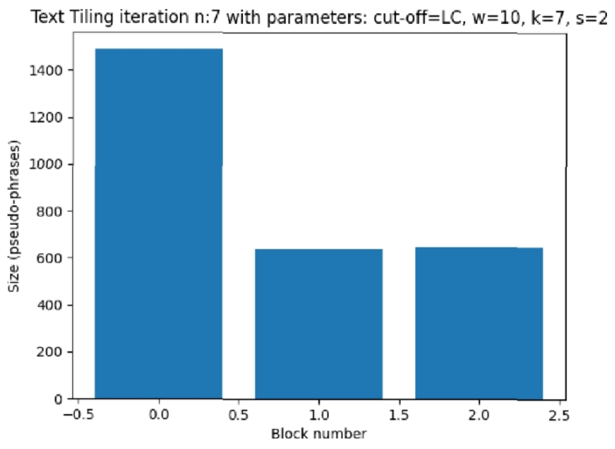

Text Tiling iteration n:5 with parameters: cut-off=LC, w=10, k=7, s=2
original
smoothed
EndOfBlock
Depth-score
Block number


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


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


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

Analysis of results

It was possible to find that the precision and recall values obtained for the LC and HC metrics (in the cases where the F1-score was the maximum) were comparable to those reported in the paper cited for the "Stargazer" paper divided into 21 paragraphs. [4]

It was also observed that the use of the HC metric produced higher values of recall the expense of precision, as indicated within the paper.

Sources

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.