Measuring Cross-Text Cohesion for Segmentation Similarity Scoring

Gerardo Ocampo Diaz and Jessica Ouyang
Department of Computer Science
University of Texas at Dallas
Richardson, TX 75080
{godiaz,jessica.ouyang}@utdallas.edu

Abstract

Text segmentation is the task of dividing a sequence of text elements (e.g., words, sentences, or paragraphs) into meaningful chunks. Although exciting advances are being made in modern segmentation-based tasks, such as automatically generating podcast chapters, current segmentation similarity metrics share a critical weakness: they are content-agnostic. In this paper, we present a word-embedding-based metric of cross-textual cohesion based on the formal linguistic definition of cohesion and incorporate it into a new segmentation similarity metric, SED. Our similarity metric, SED, is capable of providing fine-grained segmentation similarity scoring for the 3 basic segmentation errors: transposition, insertion, and deletion, as well as mixtures of them, avoiding the limitations of traditional metrics. We discuss the benefits of SED and evaluate its alignment with human judgement for each of the 3 basic error types. We show that our metric aligns with human evaluations significantly more than traditional metrics. We briefly discuss future work, such as the integration of anaphora resolution into our cohesion-based metric, and make our code publicly available.

Keywords: Segmentation, Cohesion, Segmentation Similarity, Metric

1. Introduction

Text segmentation is the task of dividing a sequence of text elements (e.g. words, sentences, or paragraphs) into segments. Formally, given a sequence of text elements $T = e_1, e_2, e_3 ... e_n$, a segmentation S of T can be defined by a binary string $Q = [0|1]^{n-1}$ that encodes boundaries between the elements of T. The ith character of Q codifies the presence of a boundary (1) or lack thereof (0) between e_i and e_{i+1} in S. S contains $m-1$ boundaries and partitions T into m segments[1].

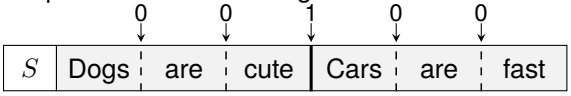


Figure 1: Example segmentation with $Q = 00100$.

While many segmentation-based tasks, such as topic segmentation, have received significant attention from researchers over the years, little progress has been since the early 2000s on improving segmentation similarity metrics. Exciting advances are being made in modern topic segmentation-based tasks, such as meeting transcript segmentation and the automatic generation of podcast chapters, but researchers continue to evaluate segmentation models using similarity metrics that share a critical weakness: they are agnostic to segment content.

[1]This definition corresponds to *single-type* segmentation. A *multi-type* version also exists where different boundary types are considered, enabling the encoding of different types of segments and even hierarchical relations between them.

Content agnosticism frequently results in segment similarity scores that are at odds with human judgment. For example, consider the task of dividing a news article into sections. In Figure 2, the reference segmentation r outlines an article covering a report on homelessness in the US: after a brief introduction, the article describes the current state of homelessness in cities across the US, then in states across the US, and concludes with a section on how the deputy secretary of the US Department of Housing and Urban Development resigned after the release of the report.

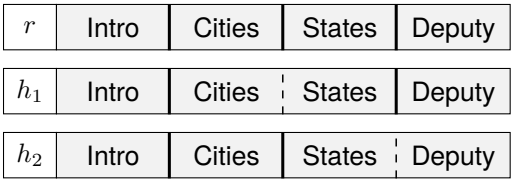


Figure 2: Reference article section outline and two alternative segmentations.

The figure shows two alternative segmentations, h_1 and h_2, which each miss a single section boundary, merging two sections from r. Traditional similarity metrics evaluate h_1 and h_2 as equidistant to r. However, human judges can perceive that h_1 is closer to r because the sections it merges are more closely related than those merged by h_2.

Current metrics cannot make this distinction because the majority — WindowDiff (Pevzner and Hearst, 2002), P_k, (Beeferman et al., 1999), and Boundary Similarity, (Fournier and Inkpen, 2012) — evaluate segmentation similarity based only on in-

LREC-COLING 2024, pages 11138–11147
20-25 May, 2024.

dividual boundary differences. The one metric that does not — A (Ocampo Diaz and Ouyang, 2022) — is based on the number, but not the content, of overlapping text elements between segments.

To address this problem, we propose a new metric, SED, that uses static word embedding similarity as a proxy for *textual cohesion*, based on definitions from linguistic theory (Halliday and Hasan, 2014). We discuss the theoretical benefits of this new metric and evaluate its correlation with human judgments. As we show in Section 4, SED aligns more closely to human judgements than existing similarity metrics. We also comment on potential avenues for future research and make our code and resources publicly available.

2. Existing Metrics

Existing segmentation similarity metrics fall into three categories[2]: **window-based** metrics slide a window across the sequence of elements, comparing at each position whether the boundaries in two segmentations match; **edit-based** metrics find a sequence of boundary edit operations that make the two segmentations equal; and **alignment-based** metrics produce a score based on segment-to-segment alignments.

Existing are based on three types of basic errors:

- **Boundary Insertion**: An extra boundary is inserted, creating two segments where one should be.
- **Boundary Deletion**: A boundary is deleted, merging two segments.
- **Boundary Transposition**: A boundary is "pushed" left or right, slightly distorting two contiguous segments.

2.1. Window-Based Metrics

P_k (Beeferman et al., 1999) and WindowDiff (Pevzner and Hearst, 2002) are the most popular similarity metrics currently used. Both slide a window of size $k+1$ across the sequence of elements and calculate a penalty based on the number of positions where the elements on the edges of the window are segmented differently by the two candidate segmentations (Figure 3).

$\mathbf{P_k}$ is defined as "the probability that a random pair of elements, k elements apart, will be classified inconsistently by two segmentations as belonging/not belonging in the same segment." Given an

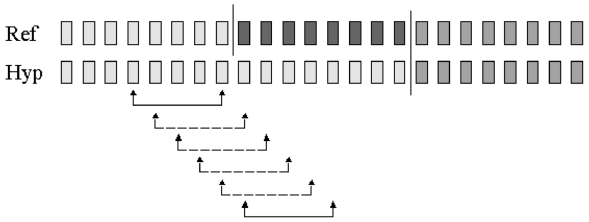


Figure 3: Illustration of P_k and WindowDiff with $k = 4$ from (Pevzner and Hearst, 2002). Penalized windows indicated by dashed lines.

element sequence T of length n, a reference segmentation r, and an alternative segmentation h, a window of size $k+1$ is slid across the elements (k is recommended to be half the average segment size in r); at each position, the r and h are compared based on the elements at the edges of the window, e_i and e_{i+k}; if the segmentations disagree on whether the elements are in the same segment, a penalty of 1 is added; finally, the penalty sum is divided by the number of windows:

$$P_k(r,h) = \frac{1}{n-k} \sum_{i=1,j=i+k}^{i=n-k} \delta(r_{i,j}) \neq \delta(h_{i,j})$$

where $\delta(x_{i,j})$ is true iff e_i, e_j are in the same segment in segmentation x.

There are a variety of situations where P_k penalizes errors inconsistently (Pevzner and Hearst, 2002): it penalizes missing boundaries more than extra boundaries, fails to penalize extra boundaries that are in close proximity to correct boundaries, and is very sensitive to the window size k.

WindowDiff (WD) improves on P_k by using a different penalty criteria. Instead of comparing the elements at the window edges, WindowDiff counts the number of boundaries between the edge elements and assigns a penalty of 1 if the number is inconsistent between segmentations:

$$\text{WD}(r,h) = \frac{1}{n-k} \sum_{i=1,j=i+k}^{i=n-k} b(r_{i,j}) \neq b(h_{i,j})$$

where $b(x_{i,j})$ is the boundary count between e_i and e_j in segmentation x.

Although WD solves some of P_ks problems, it still produces unintuitive scores and penalizes errors at the edges of the sequence less than those towards the middle. WD has been shown to produce erratic scores, for example by prematurely assigning similarity scores of 0 (Fournier and Inkpen, 2012) or by penalizing slightly transposed boundaries as harshly as missing boundaries (Fournier, 2013). Further, Ocampo Diaz and Ouyang (2022) recently showed that WD ignores the relative impact of transpositions when considering segments of different

[2]Due to length constraints, we discuss only a representative set of existing metrics; we refer the reader to the survey in Ocampo Diaz and Ouyang (2022) for a more detailed discussion of metrics and variations.

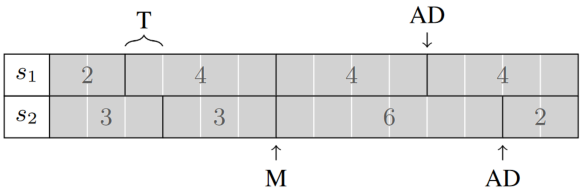


Figure 4: Example segmentation alignment with boundary edit operations from Fournier (2013).

sizes and penalizes offset boundaries equally even if they cross over into a different segment.

Although variations of the metric have been proposed (e.g. (Lamprier et al., 2007) proposes a correction to avoid the edge problem described above), the original WD metric continues to be the most used by researchers(Lukasik et al., 2020; Solbiati et al., 2021; Liu et al., 2022).

2.2. Edit-Based Metrics

Edit-based metrics are based on the Damerau-Levenshtein string edit distance (Damerau, 1964; Levenshtein, 1966) and partially replicated by Generalized Hamming Distance (Bookstein et al., 2002); they represent a segmentation as a sequence of boundaries and define a set of edit operations over that sequence. The distance between two segmentations can be measured as the cost of the optimal sequence of edit operations required to make the two segmentations equal, and the optimal sequence of edit operations is equivalent to a boundary alignment between the segmentations. Segmentation Similarity and Boundary Similarity both use the same set of edit operations (Figure 4):

- **Match:** Mark a boundary as correct.
- **Addition/Deletion:** Insert/delete a boundary.
- k**-transposition:** Shift a boundary by a max of k units. Default $k = 1$[3].

Segmentation Similarity (S) (Fournier and Inkpen, 2012) assigns a constant cost to all edit operations and normalizes the resulting distance based on the total number of possible boundaries for the given element sequence. The idea behind this normalization is to scale the cost based on the potential complexity of the segmentation in question; the intuition is that a constant cost is less impactful on a longer/more complex sequence than it is on a shorter/simpler one.

Let A_e, T_e S_e be the sets of the optimal boundary addition/deletion, transposition, and substitution operations required to align a pair of segmentations, h_1 and h_2, over a sequence of elements T. Further, let b be the number of boundary types (in the case of multi-type segmentation) available.

[3]When $k > 1$, S and B allow transpositions across existing boundaries.

$$S(h_1, h_2, T) = 1 - \frac{|A_e| + |T_e| + |S_e|}{b(|T| - 1)}$$

Fournier and Inkpen argue that S 1) produces scores that align favorably with human intuition in three key examples, 2) has reduced sensitivity to variations in segment sizes, and 3) produces more accurate inter-annotator agreement scores on one dataset, all in comparison to WindowDiff.

B improves S by introducing weighted-cost transpositions and improving the edit distance normalization factor. It is defined as

$$B(h_1, h_2, T) = 1 - \frac{|A_e| + t(T_e, k) + s(S_e, B_t)}{|A_e| + |T_e| + |S_e| + |M|}$$

where k is the maximum transposition distance, $|M|$ is the number of matching boundary pairs between h_1 and h_2, B_t is the set of boundary types, and t and s are functions that return the weighted sums of T_e (transpositions) and S_e (substitutions). The normalization factor in B produces behavior that aligns more closely with human judgement than S.

Fournier argues that B produces behavior that falls more in line with human intuition and overcomes WD's bias towards segmentations with few or tightly-clustered boundaries. However, Ocampo Diaz and Ouyang (2022) show that B also shares weaknesses with WD: it allows transpositions across segments, uses constant cost operations that do not account for the sizes of segments (e.g. a boundary offset by one element is penalized the same regardless of the lengths of the surrounding segments), and produces constant scores when boundaries are transposed beyond its maximum transposition distance.

2.3. Alignment-Based Metrics

Segment Alignment (A) (Ocampo Diaz and Ouyang, 2022) avoids the issues of window-based and edit-based similarity metrics by working at the segment level. Two segmentations h_1, h_2 are scored in two stages. First, a segment-to-segment alignment is found between h_1 and h_2 by aligning each segment to its closest segment in the other segmentation (by default, a simple intersection ratio function is used). Second, alignment edges are weighted using the Jaccard index and averaged to produce a final score.

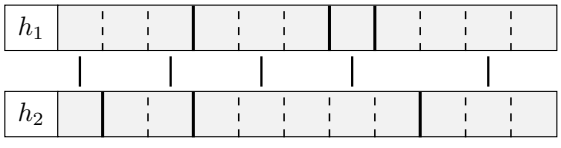


Figure 5: Example alignment produced by A. $A(h_1, h_2) = 0.51$

Figure 5 shows an example alignment generated by A. The similarity score is calculated as:

$$A(h_1, h_2, c, g) = \frac{\sum_{\text{edge} \in \text{MLA}(h_1, h_2, c)} g(\text{edge})}{\text{\# edges in MLA}(h_1, h_2, c)}$$

where MLA(h_1, h_2, c) is the function that produces the alignment (based on the intersect ratio function c) and the edge weighting function, g, is the Jaccard index between two segments, s, t:

$$J(s,t) = \frac{|\text{intersect}(s,t)|}{|\text{union}(s,t)|}$$

Note that the Jaccard edge weights are based on element indices, not on segment content.

The main benefit of A over the previous metrics discussed in this section is that it enables errors to be weighted relative to the size of the segments involved. Further, because it does not contain a maximum range for any kind of error, it does not saturate scores like WD and B. Because it works at the segment level, it does not allow transpositions across boundaries. Finally, by replacing the edge weighting function (e.g., with a semantic similarity score between aligned segments), A can be modified to to be content-sensitive. However, as we discuss in the next section, A has significant limitations of its own that we must first address.

2.4. Limitations of A

Lack of 1-to-many alignments. It is not possible for A with default settings to distinguish between alignments where segments are aligned one-to-many. Ocampo Diaz and Ouyang present the example in Figure 6 below.

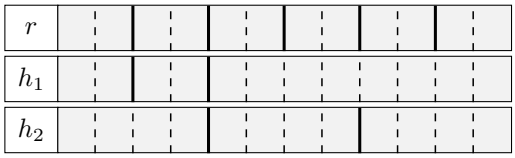


Figure 6: Reference r and two candidate segmentations h_1, h_2 that are indistinguishable using A.

Is it better for a segmenter to make many mistakes (in this case, deletions) in the same segment, as in h_1, or for the mistakes to be spread out, as in h_2? The answer may very well be application-specific, but under A, edges are weighted individually, so such distinctions cannot be made.

Boundary insertion bias. A penalizes boundary deletions more harshly than insertions. Figure 7 shows a reference segmentation r and two candidate segmentations h_1, h_2. Both candidates contain one error: h_1 deletes a boundary from r, while h_2 inserts an extra boundary; thus, both candidates can be argued to 50% correct: h_1 produces one correct segment and one incorrect segment, while h_2 produces two correct and two incorrect segments. However, A produces a score of $2/3$ for h_1 and $3/4$ for h_2. It is worth noting that the earlier metrics B and WD do not exhibit this bias.

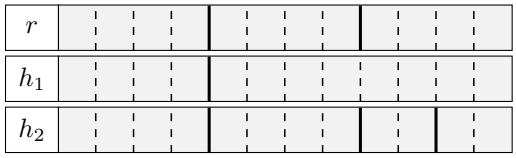


Figure 7: Insertion bias in A where candidate h_1 is scored as more similar to reference r.

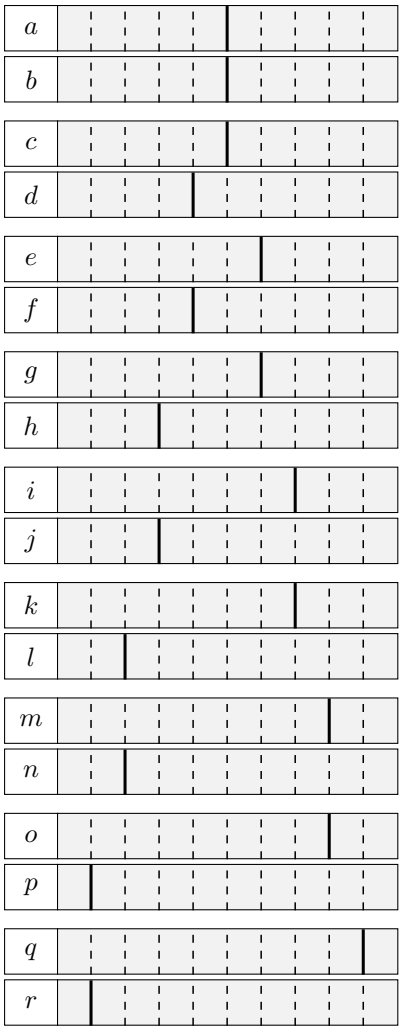
Transposition non-linearity. A does not penalize transpositions in a linear fashion. Consider the sequence of increasingly dissimilar segmentation pairs in Figure 8. As the distance between the boundaries increases linearly, we would expect the similarity score between pairs to decrease linearly, but A does not follow this trend.

It makes intuitive sense for the top left segment to align to the bottom left segment, and for the top right segment to align to the bottom right segment, but starting at pair (i, j), A allows the top-left segment to also align across to the bottom-right segment, creating an alignment with a third, diagonal edge. This cross-alignment edge receives higher and higher weights as the middle boundaries are pushed farther apart, which slows down the decrease in similarity scores.

Maximal segmentation. A penalizes extra boundaries in the range $[1, 1/n]$ (where n is the sequence length), instead of in the range $[1, 0]$. Consider the pair of segmentations in Figure 9: the reference r consists of only one large segment, while the candidate h_1 has split each element into a separate segment. WD and B assign this pair a score of 0, but A assigns a score of $1/7$. Because A weights alignment edges independently, it cannot account for the fact that h_1 is the worst possible candidate to r.

To illustrate why this is a problem, consider Figure 10 below. Both candidates h_1, h_2 insert an extra boundary, but h_1 does so in a large segment that contains many potential boundary positions, while h_2 does so in a very small segment, where there is only one possible boundary position.

Is h_1 better, or h_2? In terms of errors relative to segment size, which is the intended purpose of A, h_2's error is worse because it has failed to match the second segment of r in the worst way possible. Again, because A produces individual edges, this



Pair	A	E
(a, b)	1	1
(c, d)	0.816	0.816
(e, f)	0.666	0.666
(g, h)	0.535	0.535
(i, j)	0.419	0.428
(k, l)	0.386	0.330
(m, n)	0.366	0.250
(o, p)	0.349	0.173
(q, r)	0.340	0.111

Figure 8: Non-linear transposition penalties in A where the segmentation pairs grow more dissimilar linearly, but their scores do not decrease linearly.

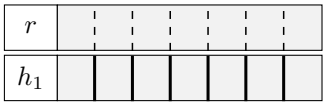


Figure 9: Maximal segmentation where h_1 is the worst possible candidate segmentation relative to the reference r; at every possible boundary position, it makes the opposite decision.

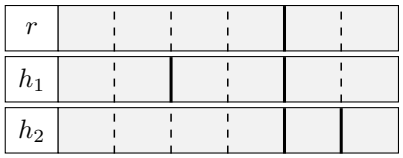


Figure 10: Reference r and candidate segmentations h_1, h_2. Both candidates make a single insertion error, but h_2's error produces a maximal segmentation of the second segment in r.

cannot be accounted for: both candidate segmentations are given a score of $2/3$.

Content Agnosticism A operates based on segment indices, not on segment content. For example, given equal length segments in figure 1, A would judge h_1 and h_2 as being equidistant to r.

3. A Global Alignment and Cohesion-Based Segmentation Similarity Metric

The weaknesses of window- and edit-based metrics stem from the fact that they operate on boundaries, not segments; while A does operate on segments, it produces and weights alignment edges individually. Finally, all current metrics are agnostic to segment content. We propose to generate a *global* segment alignment, and then weight its edges it based on the content of its segments.

3.1. Global Segment Alignments

Our new metric, *Segment Edit Distance* (SED), measures similarity as the cost of editing a reference segmentation r into a candidate segmentation h using the following operations:

- **Match/Transpose (1:1):** Match/transpose a segment from r with one from h.
- **Split (1:M):** Match/transpose a segment from r with many from h.
- **Merge (M:1):** Match/transpose many segments from r into a single segment from h.

The SED between segmentations r and h is defined as the average cost of the optimal sequence of segment edit operations. It can be found in $O(nm)$ time, where n and m are the number of segments in r and h, respectively[4].

SED effectively aligns segmentation chunks between a reference and a candidate segmentation, allowing us to penalize different types of chunks differently (1:1,1:M,M:1).

Given a reference segmentation r and a candidate h, with m and n segments, respectively, SED

[4]A python-based implementation is available at https://github.com/sierra98x/resources.

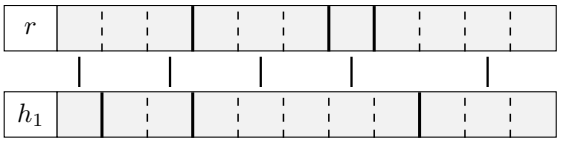


Figure 11: Example alignment produced by SED. There are three aligned chunks: (r_0-$h_{0,2}$), ($r_{1,2}$-h_2), (r_3-h_3).

can be calculated in $O(m \cdot n)$ using dynamic programming in a way similar to string edit distance. Let $cost(i, j)$ be the cost of aligning the first i segments in r with the first j segments in h. We can find $cost(i, j)$ by considering only three options based on the rightmost intersecting unaligned segments, r_i and h_j.

- Option A) (1:1) Align r_i to h_j. Then the total alignment cost is $cost(i-1, j-1)$ + cost of aligning r_i to h_j.
- If r_i reaches farther left than h_j:
  - Option B) (1:M) Align r_i to $h_{x...j}$, the longest sequence of segments in $h_{1...j}$ that intersects with r_i. Then the total cost is $cost(i-1, x-1)$ + cost of aligning r_i to $h_{x...j}$.
  - Option C) (1:M) Align r_i to $h_{x+1...j}$. Then the total cost is $cost(i-1, x)$ + cost of aligning r_i to $h_{x+1...j}$.
- Else, if h_j reaches farther left than r_i:
  - Option B) (M:1) Align h_j to $r_{x...i}$, the longest sequence of segments in $r_{1...i}$ that intersects with h_j. Then the total cost is $cost(x-1, j-1)$ + cost of aligning h_j to $r_{x...i}$.
  - Option C) (M:1) Align h_j to $r_{x+1...i}$. Then the total cost is $cost(x, j-1)$ + cost of aligning h_j to $r_{x+1...i}$.
- Finally, $cost(i, j) = min(A, B, C)$
- Note that cost(0,0) = 0 and cost(x,y) = inf if r[x] and h[y] do not intersect.

We do not need to consider more than two options for (1:M) and (M:1) alignments at each step, because all other options generate orphaned segments in r or h that have no intersecting segment to align to, invalidating the final alignment. SED(m,n) is calculated as:

Note that, unlike in A, cross-alignments (Figure 8) can not occur, since every alignment operation requires relevant segments to intersect. Also, since SED defines split and merge operations that produce alignment chunks (1:M or M:1), it can score

```
cost(R,r,h,i,j):
  if i==0 and j==0 return 0
  if r[i] and h[j] do not intersect
    return inf

  t = R[i-1][j-1] + TC(r[i],h[j])
  s1, s2 = inf

  if r[i] reachest left more than h[j]:
    z = leftmost h-seg intersecting r[i]
    s1 = R[i-1][z-1] + SC(r[i],h[z...j])
    s2 = R[i-1][z] + SC(r[i],h[z+1...j])

  if h[j] reachest left more than r[i]:
    z = leftmost r-seg intersecting h[j]
    s1 = R[z-1][j-1] + MC(r[z...i],h[j])
    s2 = R[z][j-1] + MC(r[z+1...i],h[j])

  return min(t,s1,s2)

SED(r,h):
  m,n = [r.length,h.length]
  R = m*n matrix initialized with 0s
  for i in [1,m]:
    for j in [1,n]:
      R[i][j] = cost(R,r,h,i,j)
```

Figure 12: SED pseudocode

segmentations based on *how well* a split/merge is performed: we can define cost functions that account for the complexity of the merge/split. For example, we can write cost functions that prefer balanced splits, where segments are roughly the same size; we can consider how segments in a split/merge compare to each other (Figure 2); or we could punish a segment being split into multiple smaller segments exponentially or linearly (Figure 6), relative to the worst possible merge/split.

SED also solves the insertion bias problem: because inserting/deleting additional boundaries does not change the number of edit operations (e.g., a segment being Split into three candidate segments, rather than two, is still just a single Split), we can use symmetric cost functions for splits and merges. Finally, because the edit operations are based on segment chunks, we can define different content-based penalty functions for each type of error (split,merge, transpose), something not possible in A, since it only produces alignments between individual segments.

3.2. Segmentation and Textual Cohesion

A good segmentation produces cohesive segments, so textual cohesion is an intuitive measure of segmentation quality. Textual cohesion has the benefit of being well-defined linguistically: Halliday and Hasan (2014) describe five devices used to generate cohesion in written texts:

- **Reference:** Two linguistic elements refer to the same thing. E.g., anaphora: *Jan lives by the pub. He often visits.*
- **Substitution:** One linguistic item replaces another. E.g., *A: I will have toast, please. B: I will have the same*.
- **Ellipsis:** Deletion of words or phrases, which are referenced implicitly. E.g., *A: Do you want another soda? B: Yes, another [soda] please.*
- **Conjunction:** The use of conjunctive markers such as *and, or, but, ...*
- **Lexical Cohesion:** The use of vocabulary which is related to each other either by repetition or other means. E.g., *The laundry is not done yet. I hate washing clothes*.

Paltridge and Burton (2000) further break down the different types of lexical cohesion: *repetition, synonymy, antonymy, meronymy*, and *collocation*.

Measuring Lexical Cohesion Under an edit-distance based framework, we need to measure cohesion between different segments, so not all five devices of lexical cohesion are applicable. For example, it is highly unlikely that anaphora or ellipsis would exist across separate podcast chapters or article sections. Lexical cohesion can reasonably be expected occur across segments, so we use it as a proxy for overall textual cohesion.

Word embeddings can capture lexical cohesion to some degree, as high cosine similarity scores between embeddings are the result of *collocation* and semantic similarity. This is illustrated in Figure 2: the merged segment in h_1 is more cohesive than the one in h_2 because it contains words with higher embedding similarity ("cities" and "states").

3.3. Error Penalization Criteria

Given two segments, A and B, the cross-segment lexical cohesion, $CL(A, B)$ is calculated as follows:

1. Align A and B by finding, for each token, the closest token in the opposite segment, based on static word embedding cosine similarity.
2. Calculate the directed cohesion from A to B as the sum of cosine similarities for each token in A with value $>= 0.7$[5], normalized by the total number of tokens in A.
3. Repeat for B.
4. The cross-segment lexical cohesion is the harmonic mean of the directed cohesions from A to B and B to A.

Given an alignment, we weight segmentation errors to penalize poor intra-segment cohesion within segments and strong inter-segment cohesion.

- **Transpositions:** In a transposition, a text unit is removed from one segment and added to another. The transposition is worse if the transposed unit is strongly cohesive with the segment it is removed from and is weakly cohesive with the segment it is added to.
- **Insertions:** A boundary insertion splits a single segment into two. The insertion is worse if it results in new segments that are strongly cohesive with each other.
- **Deletions:** A boundary deletion merges two segments into one. The deletion is worse if it merges uncohesive segments.

Match/Transposition Cost Function Let s_r be a segment from a reference segmentation r and s_h be an overlapping segment from a candidate segmentation h. Further, let s_h^e be the *extra* subsegment of s_h, which is not in s_r, and s_h^m be the subsegment of s_r that is *missing* from s_h. The transposition cost between s_i and s_j is

$$TC(s_r, s_h) = (1 - CL(s_r, s_h^e)) + CL(s_r, s_h^m)$$

where $CL(s_r, s_h^m)$ is the cohesion between the reference and the missing subsegment in s_h, and $(1 - CL(s_r, s_h^e))$ is the lack of cohesion between the reference and the extra subsegment in s_h.

Split Cost Function A straightforward cost function first measures the transposition cost between a reference segment s_r and the corresponding meta-segment $S_{i,j}$ that is the union of candidate segments involved in the split. Recall that the boundaries can be slightly offset by alternative segmentations, so the edges of s_r may not be exactly matched by the corresponding meta-segment $S_{i,j}$.

Then, the split is penalized based on the number of segments involved in $S_{i,j}$, relative to the worst possible number of such segments (i.e., the number of elements in s_r; Figure 10).

Finally, the split is further penalized based on the pair-wise cohesion between segments. Intuitively, if a large segment s_r is split into multiple, smaller segments in $s_h \in S_{i,j}$, then a split where the s_h are as different as possible is preferable; if the split segments are more similar to each other, it is worse for them to be separated.

[5]We empirically set this threshold to capture words that are similar enough that they would reasonably establish lexical cohesion between the two texts.

The final cost function is $1 - SC(s_r, S_{i,j})$:

$$SC(s_r, S_{i,j}) = CL(s_r, \bigcup S_{i,j}) \times \frac{(|s_r| - 1)(|S_{i,j}| - 1)}{(|s_r| - 1)} \times cdiff(S_{i,j})$$

where s_r is the reference segment, $S_{i,j}$ is a set of contiguous segments in the candidate h, and $cdiff$ is 1 - the average pairwise cohesion among $S_{i,j}$.

Merge Cost Function The merge cost function is the same as the split cost function, with the exception of the pair-wise segment cohesion penalty.

$$MC(S_{k,l}, s_h) = CL(s_h, \bigcup S_{k,l}) \times \frac{(|s_h| - 1)(|S_{k,l}| - 1)}{(|s_h| - 1)} \times csim(\bigcup S_{k,l})$$

where s_h is the candidate segment, $S_{k,l}$ is a set of contiguous segments in the reference r, and $csim$ is the average pairwise cohesion among $S_{k,l}$.

3.4. Comparison with BERT-like Approaches

In recent years, BERT-like metrics based on contextual embeddings, such as BERTScore(Zhang et al., 2020) and SBERT(Reimers and Gurevych, 2019), have become a popular approach to measuring text similarity. Although these metrics share similarities with our segment lexical cohesion metric, we argue they should not be used for segmentation scoring, as they are designed to measure *semantic similarity*, a stricter requirement than cohesion.

To illustrate why semantic similarity is not adequate for scoring segmentations, consider a reference segment $r = [s_1, s_2, s_3]$, where each element is a sentence, and an alternative segment $h = [s_1, s_2, s_3, s_4]$. We know that $s_1, s_2 s_3$ in r and h are identical; the question is how well s_4 "fits" with the other elements. If h is a good alternative segment, we would expect s_4 to be cohesive with s_1, s_2, s_3, but we would not require it to be highly semantically similar to s_1, s_2, s_3 because that would mean it is redundant with them.

If we simply applied, for example, BERTScore to measure the distance between r and h, it would perfectly align all tokens from s_1, s_2, s_3 in h to s_1, s_2, s_3 in r and the tokens from s_4 to tokens across s_1, s_2, s_3; the score reported would be the average cosine similarity between aligned tokens. This is problematic for two reasons: first, since s_1, s_2, s_3 are the same in h and r, their alignments would produce perfect cosine similarity scores, and second, the tokens in s_4 would be penalized based on how semantically similar they are to those in s_1, s_2, s_3, instead of based on how cohesive they are.

4. Human Evaluation

To test whether SED aligns with human judgements, we perform evaluation experiments using a dataset of articles downloaded from CNN.com. We generate candidate segmentations with errors that are different only in segment content and ask humans to choose between them. We find evaluators frequently agree with SED and are rarely agnostic to article content.

4.1. Data Collection

We download articles from the top five CNN categories between 2011 and 2021: *politics, us, health, entertainment,* and *opinions*. We filter out articles that 1) have fewer than four sections (based on the number of headers in the text), and 2) have a more than 1000 words (to avoid annotator fatigue). We further filter articles that contain sections shorter than 50 words, as these tend to be non-article documents, such as interviews, and we manually filter list-like documents, as they involve self-contained, unrelated sections that annotators find confusing.

4.2. Experimental Setup

For each article, we define a reference segmentation r over its paragraphs based on the section headers in the article (we consider the headers themselves to be segment boundaries, not part of the article content). We then generate distinct pairs of alternative segmentations by introducing one of the basic segmentation errors[6]: deleting a boundary, inserting an extra boundary, or transposing a boundary. Deletion pairs each delete a single (different) boundary from r; insertion pairs each insert a single boundary into the same segment of r, but in different positions; and transposition pairs each transpose the same boundary in r in opposite directions by 1 unit (paragraph). For each article and error type, we select a pair h_1, h_2 to maximize the difference $|SED(r, h_1) - SED(r, h_2)|$[7], with the

[6]We do not produce errors involving the first section; the first section is frequently an introduction and is thus semantically different from the other sections.

[7]We perform tokenization, lemmatization, POS tagging, filtering, and word vector calculations using the spaCy `en_core_web_lg` v 3.7.1 pipeline. We filter any words that are not nouns, verbs, adjectives, or adverbs.

goal of testing whether humans agree that there is a difference between them.

For each error type, we present three human judges with 30 articles[8], each paired with two segmentations h_1,h_2, and ask them whether h_1 is better, h_2 is better, or h_1, h_2 are equally good/indistinguishable, taking the majority vote among judges.

4.3. Results

For each basic error, we report three values: inter-annotator agreement (Fleiss multi-$/kappa$), percentage agreement between SED and human majority vote, and percentage agreement between prior metrics (i.e., judging all pairs to be indistinguishable) and human majority vote.

Error	Fleiss κ	SED %	Prior %
Deletion	0.57	0.87	0.06
Insertion	0.50	0.60	0.03
Transposition	0.19	0.44	0.40

Table 1: Human evaluation results for the three types of basic errors.

The cohesion-based penalization used by SED works particularly well for deletion errors, with human judges reporting high levels of chance-corrected agreement (0.57) and metric accuracy (0.87%) against human majority vote. Human judges have similarly high agreement for insertion errors (0.5), but here SED's accuracy against the human majority vote is only moderate (0.60). Finally, both human agreement and accuracy against the majority vote are lowest for transposition errors (0.19, 0.44, respectively).

The level of difficulty reported by judges for each error type aligns well with the inter-annotator agreement results. Judges rank deletions as being the easiest to evaluate, then insertions, and finally, transpositions as the hardest:

- Deletion errors are the easiest to judge because they involve merging two gold segments (each on one side of the deleted boundary) that are both semantically meaningful and distinct.
  - Example: In an article about dogs, h_1 may merge the topics "food" and "treats", while h_2 merges "discipline" and "health". Because all the segments are semantically meaningful and sufficiently distinct, it is easy for human judges to decide which merging is worse.
- Insertion errors are harder because they split a gold segment with no guarantee that the resulting halves will be semantically meaningful or distinct.
  - Example: In the same article about dogs, the "food" section is split by inserting an extra boundary, but the "food" section talks about different foods that are good for dogs in no particular order, so it becomes hard for evaluators compare two segmentations h_1,h_2 that split "food" in different ways.
- Transposition errors are the hardest because they involve the first and last paragraphs of neighboring sections, which tend to be equally heavy in terms of topic-specific content; both candidate segmentations look equally bad to human judges.
  - Example: Given two contiguous sections in r, "dogs" and "cats", h_1 expands "dogs" so that it now includes the first paragraph of "cats", while h_2 expands "cats" so that it now includes the last paragraph of "dogs" — both are bad.

Overall, the content-agnostic approach of prior metrics does not align well with human judgements. For deletions and insertions, humans rarely judge pairs as being equal. We see an increase in humans selecting ties for transpositions, although this could be attributed to the difficulty of the task.

5. Limitations

Like all segmentation similarity metrics, our work assumes segment homogeneity; for example, in Figure 2, SED would correctly identify that merging the *City* section with the *States* section is better than merging the *States* section with the *Deputy* section, but things are less clear when merging the *Intro* section with the *Cities* section, as the *Intro* section plays a very different role in the article than the other sections do. There is no straightforward solution to this problem, as *segment types* are frequently domain-specific.

6. Conclusions and Future Work

Textual cohesion enables segmentation similarity metrics to move towards content-based scoring. Our experiments show that SED, with a simple word-embedding-based metric of cross-text cohesion, aligns significantly well with human evaluations of segmentation similarity, unlike traditional metrics.

There are two straightforward research directions in which we are interested: first, the incorporation of anaphora resolution into our cohesion metric as a pre-processing step would help capture more cases of lexical cohesion; second, it is relatively simple to modify the cross-text cohesion metric to measure inner-text cohesion, which can be used as a proxy for segment quality.

We make our code and data available for public use at https://github.com/sierra98x/resources.

[8]We limit the number of articles to prevent fatigue, as judges must read large sections of the articles.

7. References

Doug Beeferman, Adam Berger, and John Lafferty. 1999. Statistical models for text segmentation. *Machine learning*, 34(1):177–210.

Abraham Bookstein, Vladimir A. Kulyukin, and Timo Raita. 2002. Generalized hamming distance. *Inf. Retr.*, 5(4):353–375.

Fred J. Damerau. 1964. A technique for computer detection and correction of spelling errors. *Commun. ACM*, 7(3):171–176.

Chris Fournier. 2013. Evaluating text segmentation using boundary edit distance. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1702–1712, Sofia, Bulgaria. Association for Computational Linguistics.

Chris Fournier and Diana Inkpen. 2012. Segmentation similarity and agreement. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 152–161, Montréal, Canada. Association for Computational Linguistics.

Michael Alexander Kirkwood Halliday and Ruqaiya Hasan. 2014. *Cohesion in english*. 9. Routledge.

Sylvain Lamprier, Tassadit Amghar, Bernard Levrat, and Frederic Saubion. 2007. On evaluation methodologies for text segmentation algorithms. In *19th IEEE International Conference on Tools with Artificial Intelligence(ICTAI 2007)*, volume 2, pages 19–26.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet Physics Doklady*, volume 10, page 707.

Yang Liu, Chenguang Zhu, and Michael Zeng. 2022. End-to-end segmentation-based news summarization. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 544–554.

Michal Lukasik, Boris Dadachev, Kishore Papineni, and Gonçalo Simões. 2020. Text segmentation by cross segment attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4707–4716.

Gerardo Ocampo Diaz and Jessica Ouyang. 2022. An alignment-based approach to text segmentation similarity scoring. In *Proceedings of the 26th Conference on Computational Natural Language Learning (CoNLL)*, pages 374–383, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Brian Paltridge and Jill Burton. 2000. *Making sense of discourse analysis*. Antipodean Educational Enterprises Gold Coast, Queensland.

Lev Pevzner and Marti A. Hearst. 2002. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28(1):19–36.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992.

Alessandro Solbiati, Kevin Heffernan, Georgios Damaskinos, Shivani Poddar, Shubham Modi, and Jacques Cali. 2021. Unsupervised topic segmentation of meetings with bert embeddings. *arXiv e-prints*, pages arXiv–2106.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.