Text segmentation similarity revisited: A flexible distance-based approach for multiple boundary types

Ryan Ka Yau Lai, Yujie Li
University of California, Santa Barbara
{kayaulai, yujie_li}@ucsb.edu

Shujie Zhang
University of California, Berkeley
z4362687@berkeley.edu

Abstract

Segmentation of texts into discourse and prosodic units is a ubiquitous problem in corpus linguistics and psycholinguistics, yet best practices for its evaluation – whether evaluating consistency between human segmenters or humanlikeness of machine segmenters – remain understudied. Building on segmentation edit distance (Fournier & Inkpen 2012, Fournier 2013), this paper introduces a new measure for evaluating similarity between two segmentations of the same text with multiple, mutually exclusive boundary types, accounting for varying identifiability and confusability between these types. We implement a dynamic programming algorithm for calculation specifically geared towards this type of segmentation problem, apply it to a case study of intonation unit segmentation measuring inter-annotator agreement, and make suggestions for interpreting results.

1 Introduction

In computational corpus linguistics and psycholinguistics, many types of annotation and experimental tasks can be seen as *segmentation* problems, where a text is broken up into segments. These segments can be morphemes, tokens (i.e. tokenisation), prosodic, syntactic and interactional units (such as intonation units, sentences, utterances and turns), as well as larger segments of discourse like topics.

When multiple annotators, whether human or machine, have annotated the same text, the question arises as to how to measure the degree of divergence. There are multiple motivations for this question. Methodologically, we often want to evaluate annotation schemes and annotator training (e.g. Lin 2009), as well as humanlikeness of computational segmentation models. Theoretically, comparing the consistency of different types of segmentation sheds light on human perception of boundaries, such as how boundaries are perceived (e.g. Troiani et al. 2023).

This paper focuses on one type of problem: segmentation using a set of mutually exclusive boundary types. Punctuation prediction (Lu & Ng 2010), for example, can be seen as this task: a text is divided using a set of mutually exclusive punctuation marks. Consider, as an example, the following unpunctuated, 5-word text:

London Bridge is falling down

We assume that each word is potentially followed by a punctuation mark, so there are 5 possible spots to place boundaries. Under this situation, there cannot be more than one punctuation mark between two words (unlike an application where, for example, one segments a text into both sentences and paragraphs, and hence a space may be both a sentence boundary and a paragraph boundary). Assuming there are three candidate punctuations, comma, period and question mark, the sets of choices are thus (Ø represents no boundary):

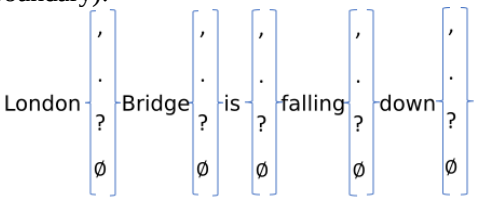

Figure 1: Schematic illustration of the type of segmentation problem explored in this paper, where each potential boundary can be one of a fixed set of mutually exclusive boundary types.

In the computational literature, various metrics for evaluating segmentation differences have been proposed and examined (e.g. Beeferman et al.

Proceedings of the Society for Computation in Linguistics (SCiL) 2023, pages 300-309.
Amherst, Massachusetts, June 15-17, 2023

1999, Pevzner & Hearst 2002, Lamprier et al. 2007, Franz et al. 2007, Peshkov et al. 2013, Peshkov & Prévot 2014). To our knowledge, however, none are specifically geared towards this type of problem. An additional complication of this paper is that our method must work for both monologic and dialogic texts, which none of the previous methods have focused on.

In this paper, building on Fournier & Inkpen (2012) and Fournier (2013), we propose a new metric, *flexible segmentation similarity* (S_f), allowing not just for gradient similarities between boundary types, as discussed also by Fournier (2013), but also for differentiating insertion and deletion of different boundary types. We also discuss a simulation-based approach to calculate Cohen's κ inter-annotator agreement for this measure. We apply the method to a case study of intonation unit (IU) segmentation, where part of the NCCU Taiwan Mandarin Corpus (Chui & Lai 2008) was manually segmented into IUs, and each IU boundary was classified according to boundary intonation preceding it.

2 Previous work

Many evaluation metrics have been applied to segmentation, including match percentage (Lin 2009), conventional measures of classification performance like precision, recall, F1 value and accuracy, windows-based approaches like P_k (Beeferman et al. 1999) and WindowDiff (Pevzner & Hearst 2002), and edit distance-based methods (Fournier & Inkpen 2012, Fournier 2013). Our measure builds on the last approach due to significant disadvantages of the rest.

2.1 Problems with non-edit distance-based metrics

The pros and cons of these methods are widely discussed in the literature (e.g. Beeferman et al. 1999, Pevzner & Hearst 2002, Lamprier et al. 2007, Franz et al. 2007, Fournier & Inkpen 2012, Peshkov et al. 2013, Peshkov & Prévot 2014), but several problems stand out. Firstly, conventional classification performance measures and match percentage fail to account for 'near-misses' (Pevzner & Hearst 2002), where two annotators place the 'same' boundary in different but close locations. This is often the case in intonation unit boundary identification: different boundary-marking acoustic cues can be spread across multiple words, creating fuzzy boundaries (Barth-Weingarten 2016: 6-7). Treating the problem as simple classification unduly penalises such cases. For example, Troiani et al. (2023) find that English speakers have very poor performance on segmenting Kazakh texts into intonational boundaries when the many near-misses are ignored; this was likely due to uncertainty about which part of the recording corresponded to which part of the transcript.

Secondly, conventional classification performance measures and windows-based metrics are asymmetric (Fournier 2013): We evaluate one annotation set against another; switching the places of the two annotations results in different numbers. So these measures can only compare one annotation against a gold standard, but not when there is no ground truth (e.g. between two equally-trained human annotators).

Thirdly, all non-edit-distance-based measures do not account for multiple boundary types, which often arise in corpus linguistics, and treat all mistakes as 'equal', ignoring differences in difficulty between boundary types (cf. Qian et al 2016 in the tokenisation context).

2.2 Edit distance-based metrics

The edit distance-based approaches Segmentation Similarity (*S*) (Fournier & Inkpen 2012; henceforth F&I) and Boundary Similarity (*B*) (Fournier 2013) are the closest to our proposed measure, as they account for near-misses, are symmetric, and allow multiple boundary types. They are briefly reviewed here with our simplified notation, which will be used throughout this paper.

In the following, we will refer to the elements between which boundaries can be added as *tokens*. This may be roughly words in tasks like intonation unit segmentation, or a larger unit like turn-constructional units in turn segmentation, sentences in topic segmentation, and so on. For *S* and *B*, the number of potential boundaries *N* is the number of tokens minus 1. The potential boundaries in a text will be denoted $b_1, b_2, \ldots, b_{N-1}$; for example, in the Figure 1 example, b_1 is between *London* and *Bridge*, b_2 between *Bridge* and *is*, and so on. The actual boundaries from annotator *i* will be denoted $b_{i,1}, b_{i,2}, \ldots, b_{i,N-1}$. Since these measures deal with non-mutually exclusive boundary types, each of $b_{i,1}, b_{i,2}, \ldots, b_{i,N-1}$ is a *set* of boundaries. For example, when simultaneously annotating turn

and sentence boundaries, one potential boundary could be both a turn boundary and a sentence boundary.

For calculating *S*, one set of annotations is transformed into another, minimising the number of operations taken. There are three possible operations: a) adding boundaries, b) deleting boundaries, and c) transposing boundaries, i.e. moving a boundary to a different position, in order to align it with a boundary placed by another annotator. Thus, if one annotator put a boundary in $b_{1,1}$ but not in $b_{1,2}$, and another put a boundary of the same type in $b_{2,2}$ but not in $b_{2,1}$, then we can transpose the boundary from $b_{1,1}$ to $b_{1,2}$ to match the second annotator. This only takes one operation, as opposed to deleting in $b_{1,1}$ and adding it to $b_{1,2}$, which takes two, thus preventing the problem of overpenalising near-misses.

The similarity is then calculated thus:

$$S = \frac{N - \#(\text{edits})}{N} = 1 - \frac{\#(\text{edits})}{N}$$

S is thus a ratio in [0, 1]: the larger *S* is, the closer the annotations. F&I also mention the possibility of scaling the number of boundaries 'moved' in transposition so that e.g. 2 transpositions might count for fewer than two edits.

B differs from *S* in two ways. Firstly, the normalisation is different. The score is normalised by the number of edits plus the number of correct boundaries. This in essence means the number of total boundaries perceived by the two annotators, assuming that transposed boundaries are the 'same' boundary across annotators. This prevents biasing annotators towards a smaller number of boundaries, i.e. longer segments. This is useful for tasks like intonation unit segmentation: in languages like English and Mandarin, intonation unit boundaries in spoken language are typically denser than punctuation boundaries in written language. Annotators may be influenced more by orthography if biased towards fewer boundaries.

Secondly, instead of the number of edits, the distance between the two annotations is calculated more flexibly by assigning different costs to different edit operations. Although addition and deletion retain the cost of 1, *B* allows for substitutions between boundary types. For *B*, boundary types are organised on an ordinal scale, and the cost of substituting one boundary for another is their distance on their ordinal scale normalised by the total number of boundary types. The formula for *B* is as follows:

$$B = 1 - \frac{C_{total}}{\#(\text{edits}) + \#(\text{correct boundaries})}$$

where C_{total} is the total cost of operations.

Although *S* and *B* are excellent measures of similarity between different annotations, they still have disadvantages. Firstly, although *B* allows for different similarity between different boundary types, recognising that some boundaries may be more confusable than others, it makes the strong assumption that these differences are gradable on an ordinal scale, which is problematic for intonation unit segmentation (see Section 4).

Secondly, by setting addition and deletion cost by default to 1, it ignores the fact that some boundaries may be easier to identify than others. Deleting an easy boundary should cost more than deleting a difficult one.

Finally, *S* and *B* are excellent for written and monologic texts, but are unsuited for multi-party conversations where tokens are not organised in a single linear sequence, since two people's speech can overlap. Our proposed method addresses all three of these weaknesses.

3 Proposed method

3.1 Definition of S_f

Like Segmentation Similarity (*S*) and Boundary Similarity (*B*), we evaluate similarity first by transforming one annotation to the other and calculating the cost, normalising this, and then subtracting the similarity from 1 to get a distance. We present two options for normalising: following *S* in using the number of potential boundaries (S_f) and following *B* in using the number of edits plus correct boundaries (S_f^B) (*pace* Fournier (2013), we argue (Section 4) that both normalisation approaches can be useful in different situations):

$$S_f = 1 - \frac{C_{total}}{N}$$

$$S_f^B = 1 - \frac{C_{total}}{\#(\text{edits}) + \#(\text{correct boundaries})}$$

The calculation of C_{total} departs substantially from *S* and *B*. We allow for user-defined addition, deletion and substitution costs using the similarity matrix M_T. The values in the matrix are the similarity (on the interval [0, 1]) between the two different boundary types. One minus the value is the cost of substitution between these two boundary types. The final row and column are for the *lack* of a boundary. Here is a sample matrix with two boundary types $T = \{p, q\}$:

$$M_T = \begin{pmatrix} 1 & s_{pq} & s_{p\emptyset} \\ s_{qp} & 1 & s_{q\emptyset} \\ s_{\emptyset p} & s_{\emptyset q} & 1 \end{pmatrix}$$

Here, s_{ab} is one minus the cost of substituting a for b, and $\emptyset$ refers to the lack of a boundary. Thus, $1 - s_{\emptyset q}$ is the addition cost of q, and $1 - s_{p\emptyset}$ is the deletion cost of p. When a symmetric score is desired, e.g. comparing two human annotators, the matrix must be symmetric as well, i.e. $s_{xy} = s_{yx}\ \forall x, y \in T \cup \{\emptyset\}$. This means substituting *x* for *y* has the same cost as substituting *y* for *x*, and insertion and deletion have identical costs. By default, this matrix is the identity matrix I, i.e. all substitutions, additions and deletions have a cost of 1. An example of user-defined M_T will be given in Section 4; in cases where existing annotations by expert annotators are available, confusion matrices from those raters can be used instead to determine M_T in evaluating similarity between novice annotators' work. In the rest of this paper, addition and deletion will be treated as special cases of substitution involving $\emptyset$.

Transposition cost can also be set flexibly for different boundary types, represented by c^t, a vector with as many entries as there are boundary types. In this paper, we will set transposition cost at half of insertion/deletion cost, i.e., $c^t = \frac{1}{2}(\mathbf{1} - \left[s_{p\emptyset}\ \ s_{q\emptyset}\right]^T)$. A glossary of notation is in Table 1.

S_f , S_f^B	Flexible segmentation distance normalised respectively with *N* and $\#(edits) + \#(correct\ boundaries)$
C_{total}	Total cost of transforming between annotations
$\emptyset$	No boundary
N	Number of potential boundaries
b_i	*i*th potential boundary
$b_{i,j}$	Annotator *j*'s annotation of b_i
T	Set of boundary types
M_T	Similarity matrix for T
s_{pq}	Similarity between *p* and *q*
$t_1[x]$	*x*th element of boundary list t_1
$t_1[-x]$	t_1 without the *x*th element
$t_1[x:y]$	*x*th to *y*th elements of t_1
c^t	Vector of transposition costs
$\mathrm{tr}(t_1, x, y)$	boundary list t_1 with the *x*th and *y*th elements swapped
κ	Cohen's kappa
S_f^{chance}	Chance-level similarity

Table 1: Glossary of notation used in this paper.

3.2 Algorithm for calculating S_f

Our algorithm first separates the text by conversational participants, since tokens from the same participant cannot overlap, and thus can be taken as one whole running text. We calculate the cost for each participant separately, then take the sum. Also, in our use cases, the end of the text also has a boundary type, so number of potential boundaries *N* is equal to the number of tokens.

For each participant, we first identify all the potential boundaries where *both* annotators put a boundary, regardless of whether their types match. Our boundary types are mutually exclusive, so when two people put different boundaries in the same place, they can be safely assumed to have *identified* the 'same' boundary, and just *classified* it differently. We treat these as substitutions and store the total cost of these operations.

We then further split the text into smaller lists of boundaries at those points where both annotators have a boundary. Consider the following situation:

	b_1	b_2	b_3	b_4	b_5	b_6
Annotator 1	p		p			q
Annotator 2	q		p	p		p

We will split the text at the points b_1, b_3 and b_6, leaving two boundary lists: $[b_2]$ and $[b_4\ b_5]$.[1] We discard the boundary lists where both annotators have no boundaries since such lists are necessarily identical. In this case, we discard $[b_2]$, retaining only the single list $[b_4\ b_5]$. For the remaining boundary lists, we trim all the **common** leading and trailing $\emptyset$s in both lists, since it is pointless to move boundaries to those locations; this leaves only $[b_4]$ in the example.

We then calculate the similarity between the two annotators for each of these segments. For each segment, we run a recursive algorithm, called parDist, to find the minimum cost of transforming one annotation to the next. At each step, we first trim any common leading and trailing $\emptyset$s again. We then choose the next step depending on properties of the two boundary lists:

- If the two boundary lists have size 1, then we simply return the substitution cost (which is 0 if they are the same boundary, and >0 otherwise).

[1] We assume that annotators will not place a single boundary of indeterminate location in more than one spot; thus, Annotator 2 is committing to there being two distinct boundaries at b_4 and b_5.

- If the length is >1, we look for positions where *both* boundary lists have a boundary. If one such position exists, we perform a substitution at it, then perform parDist on the remaining contiguous portion(s) of the boundary list. For example, if the lists have five elements and this substitution happens at the fourth element, then we run parDist again on two boundary sub-lists: the first three elements and fifth element. If the substitution on this list happens at the first element, then we run parDist on the segment from the second to fifth item.
- If there are no positions where both boundary lists have a boundary, but both lists have at least one non-$\emptyset$ boundary, then we attempt both transposition and substitution and take the minimum. For transposition, we attempt to move the first non- $\emptyset$ boundary in the second boundary list so that it matches up with an element in the first boundary list, then run parDist on the resultant boundary lists. For substitution, we simply replace the first element of the second boundary list with the first element of the first boundary list, then run parDist on the remaining boundaries. We take the transposition cost if it is smaller, and vice versa.
- Finally, if one of the boundaries consists of all 0s, then we perform substitution until all the differences are eliminated.

A rough presentation of parDist in pseudocode is presented in Algorithm 1, where t_1 and t_2 are the two annotations, $t_1[1]$ refers to the first element of the boundary list, $t_1[-1]$ refers to the boundary list without the first element, $t_1[x{:}\,y]$ refers to the *x*th to *y*th elements of t_1, and $\mathrm{tr}(t_1, x, y)$ refers to t_1 with the *x*th and *y*th elements swapped. Figure 2 illustrates the algorithm with a concrete example.

function parDist(t_1, t_2):

remove all **common** leading and trailing $\emptyset$s from t_1 and t_2
if length(t_1) ≤ 1:
  return $1 - s_{t_1[1],t_2[1]}$
else:
  if $t_1[1] = t_2[1]$:
    return parDist $(t_1[-1],\ t_2[-1])$
  else if($t_1[1] \neq \emptyset$ & $t_2[1] \neq \emptyset$):
    return $1 - s_{t_1[1],t_2[1]}$
      + parDist $(t_1[-1],\ t_2[-1])$
  else if $\exists\ i$ such that $t_1[i] \neq \emptyset$ & $t_2[i] \neq \emptyset$:
    take the smallest i
    return $1 - s_{t_1[i],t_2[i]}$
      + parDist $(t_1[1{:}\,i-1],\ t_2[1{:}\,i-1])$
      + parDist $(t_1[i+1{:}\,\mathrm{length}(t_1)],$
      $t_2[i+1{:}\,\mathrm{length}(t_1)])$
  else if $t_1[1] = \emptyset$ & $\exists\ i$ such that $t_1[i] \neq \emptyset$:
    take the smallest i
    return $\min(1 - s_{t_1[1],t_2[1]}$
      $+ \mathrm{parDist}(t_1[-1],\ t_2[-1]),$
      $c^t[t_2[1]] \cdot (i-1)$
      $+ \mathrm{parDist}(t_1,\ \mathrm{tr}(t_2, 1, i)))$
  else if $t_2[1] = \emptyset$ & $\exists\ i$ such that $t_2[i] \neq \emptyset$:
    take the smallest i
    return $\min(1 - s_{t_1[1],t_2[1]}$
      $+ \mathrm{parDist}(t_1[-1],\ t_2[-1]),$
      $c^t[t_2[i]] \cdot (i-1)$
      $+ \mathrm{parDist}(t_1, \mathrm{tr}(t_2, 1, i)))$
  else:
    return $1 - s_{t_1[1],t_2[1]}$
      + parDist$(t_1[-1],\ t_2[-1])$

Algorithm 1: Pseudocode for parDist

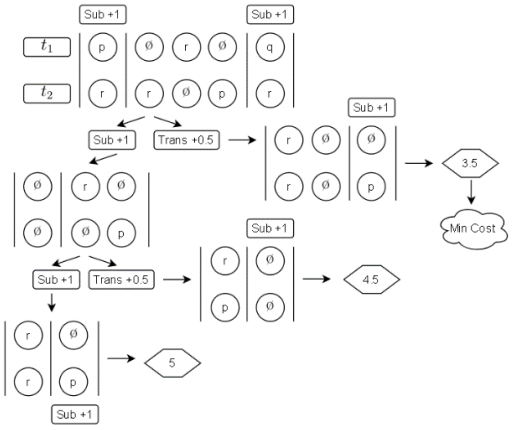

Figure 2: An illustration of parDist, assuming $M_T = I$ (i.e. substitutions including insertion and deletion cost 1), transpositions cost 0.5, and $T = \{p, q, r\}$. Firstly, all positions with a boundary in both annotations are considered substitutions. There are then two options: Either move the *r* of the second annotation to the right, or delete it. In the first case, the *p* must then be deleted, resulting in a cost of 3.5. In the second case, one can then either bring the *p* to the left then substitute it for an *r* (cost = 4.5), or add an *r* and delete the *p* (cost = 5). The minimum cost of all these possibilities is then 3.5.

The actual implementation of the algorithm involves several components omitted from the pseudocode for a cleaner presentation. Two of these components aim at storing information about the process. Firstly, the number of actions hitherto performed is stored and accumulated across iterations of parDist to calculate the

denominator of S_f^B. Secondly, information about each operation – including the operation type, old and new boundary type, and old and new location – can be stored for later access so they can be used for analysing machine segmentation errors or points of inter-annotator disagreement (Section 4 has an example).

Two other components aim at speeding up computation. Firstly, the function stores the minimum cost so far among the total costs that have been calculated. When the cumulative cost in the branch of possibilities currently being explored has exceeded the stored minimum, the function returns NA, thereby aborting the branch, instead of continuing the calculation. Secondly, every time parDist is calculated, the resulting cost and number of operations are stored in a two-dimensional dictionary with t_1 and t_2 (stored as strings) for keys. Before each instance of parDist, the algorithm looks up the dictionary and simply takes the result from there if the operation has been done before. These result in significant speed gains, especially when calculating similarity between simulated annotations for inter-annotator agreement (see Section 3.3).

A property of this algorithm is that a boundary may be both transposed and substituted if the cost of doing so is lower than insertion plus deletion. For the calculation of $\#(edits)$ in the S_f^B formula, such edits will only be counted once, in the spirit of normalising by the total number of boundaries. A consequence of this property is that the algorithm differs from one which decomposes the process of similarity calculation into a two-step process, where boundaries are first aligned ignoring boundary type, and then substitution costs are calculated. This is because substitution cost can affect whether a boundary in t_1 which corresponds to $\emptyset$ in t_2 is simply deleted, or transposed to match with a nearby boundary by the other annotator.

Our algorithm was implemented in R (R Core Team 2022). It takes input data formatted as an R `data.frame`, and outputs S_f and S_f^B, along with a record of each operation that took place. We additionally wrote a function to convert files in the .rez format imported from Rezonator (DuBois et al. 2020) into the required format for the function. The algorithm is available as an R package (https://github.com/rezonators/segsimflex).

3.3 Inter-annotator agreement

The similarity score measures how similar two annotations are, but how similar counts as 'good'? Converting the similarity to inter-annotator agreement (IAA) (Passonneau & Litman 1993, Hearst 1997) allows us to directly measure agreement among annotations. We use Cohen's κ, which compares the actual similarity between the annotations against chance-level similarity.

Following the definition of Cohen's κ, we calculate chance-level similarity based on the assumption that each boundary is a categorical random variable where each category is a boundary type or no boundary. The annotations are independent and identical within annotators and independent but non-identical across annotators. For example, with two boundary types p and q, the categories are $\{p, q, \emptyset\}$, and each annotator has their own $P(p)$, $P(q)$ and $P(\emptyset)$ values. Category probabilities are estimated with the maximum likelihood estimator, i.e. the proportion of that category within the annotation.

Based on these estimated null distributions, we then estimate chance-level similarity S_f^{chance} using the expected value of the similarity score. We use a simulation approach since it is difficult to find a closed form for it. At each simulation step, we draw a boundary type for each annotator at each boundary, then calculate the similarity score. The average similarity over k simulation steps is the estimated expected value of the similarity score. Cohen's κ is then calculated thus:

$$\kappa = \frac{S_f - S_f^{chance}}{1 - S_f^{chance}}$$

Hence, a negative score means below-chance performance, a positive score is above-chance, and perfect performance results in $\kappa = 1$.

Both S_f and S_f^B can be used for κ. If S_f is used, then the form of κ used here resembles the standard form of κ in classification tasks, except with gradient similarity between categories and an added possibility of transposition. Nevertheless, S_f may still be advisable at least in some situations (see Section 4.4 for discussion).

A common criticism of κ in classification contexts (Byrt, Bishop & Carlin 2010) is that large differences in raters' individual category distributions will deflate chance-level agreement and push κ up. In cases where this is expected to be a substantial problem, S_f^{chance} can instead be

calculated using an overall estimation of category probabilities that pools together both raters' annotations, turning the IAA into Scott's π.

Another common criticism is that a situation with unbalanced categories will lead to drastically higher expected proportion of agreement and thus lower κ values than one with balanced categories. This phenomenon is likely to occur with S_f-based κ, since non-boundaries are much more common than boundaries, but it is not necessarily problematic: A text with many non-zero boundaries is 'harder' to get right than a text with few non-zero boundaries, so if the aim is measuring rater performance (rather than the quality of the annotation itself), texts with more non-zero boundaries should have higher IAA than those with fewer non-zero boundaries but a comparable level of similarity. If the phenomenon *is* problematic, S_f^{chance} can instead be calculated based on the assumption that all boundary types (including no boundary) have equal probability, turning the IAA measure into Bennett's *S*. S_f^B-based κ ignores non-boundaries in normalising agreement, and thus is less likely to be subject to this phenomenon; if unevenness among boundary types is an issue, one may modify Bennett's *S* such that S_f^{chance} is calculated by getting a pooled estimate of the probability having no boundary from the two raters, then assuming the distribution of boundary types is uniform.

4 Case study: intonation unit segmentation

To illustrate the proposed measure, we apply our proposed measure to exploring inter-annoator agreement in a prosodic segmentation task.

4.1 Data and problem

We are manually segmenting the NCCU Taiwan Mandarin Corpus (Chui & Lai 2008) into intonational units (IUs), a unit of prosody corresponding to short bursts of speech (roughly corresponding to intonation phrases or breath groups in other prosodic frameworks). So far, we have annotated texts TM001, 004, 009, 016, 025, 036, 049. Before IU segmentation, we tokenised the texts to obtain potential boundary locations, following principles in Huang et al. (1997, 2017).

Two independent coders perform IU segmentation using four main boundary types, called *endnotes*, representing broad classes of prosodic contours near the end of the IU, each of which signals a type of transitional continuity (DuBois et al. 1993, DuBois 2020): Rising intonation indicating appeal, as in questions and uptalk (denoted by ?), continuing intonation indicating continuation of the prosodic sentence (a comma ,), falling intonation indicating finality (a period .), and a boundary marker for truncated IUs, i.e. IUs that ended before completion (a dash --). Some boundaries were uncategorised, usually because the IU consisted solely of elements with no discernable prosody, e.g. laughter or tsk-tsk; these are denoted as semicolon (;). Earlier on in the process, texts were segmented by manually editing text files; later, we performed segmentation using the Rezonator program (DuBois et al. 2020). Figure 3 shows the same tokens from one of the texts, TM001, as segmented differently by the two annotators who worked on this text. We calculate similarity scores and IAA on these texts to evaluate the quality of our annotation training and workflow and identify avenues for improvement.

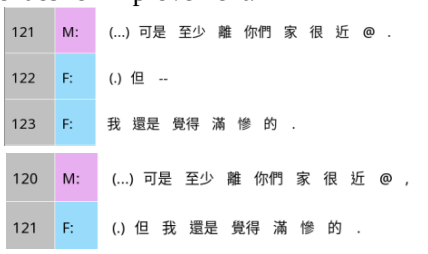

Figure 3: Example annotations in TM001. (a) and (b) are from two different annotators. The first boundary was deemed final by the first annotator, and continuing by the second. The word 但 *dàn* 'but' was put in a separate IU by the first annotator, but not the second.

4.2 Parameter values

For each pair of annotators, we calculated four values: S_f and S_f^B with an identity distance matrix, and the same values with the following custom similarity matrix:

$$M'_T = \begin{array}{c} \\ ? \\ , \\ . \\ -- \\ ; \\ \phi \end{array} \begin{array}{c} \begin{array}{cccccc} ? & , & . & -- & ; & \phi \end{array} \\ \begin{pmatrix} 1 & .5 & .25 & .25 & 1 & 0 \\ .5 & 1 & .5 & .5 & 1 & .25 \\ .25 & .5 & 1 & .25 & 1 & 0 \\ .25 & .5 & .25 & 1 & 1 & .25 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & .25 & 0 & .25 & 1 & 1 \end{pmatrix} \end{array}$$

Rising and falling intonation have the most dissimilar pitch contour of the four, hence a similarity of .25. Truncated intonation differs from

all others in not following a complete prosodic gestalt, and resembles continuing in having no rise/fall; hence the similarity with continuing is.5, and the similarity with the rest is .25. Rising and falling endnotes are substantially different intonationally from an IU-medial word, so their similarity with no boundary is 0; continuing and truncated IUs have less clear pitch cues and hence are harder to detect consistently, and receive .25 similarity. For simplicity, unclassified boundaries are ignored by treating them as identical to all other boundaries. Transposition costs are set at half the insertion/deletion cost for each endnote.

One may ask why we use these hand-crafted 'theoretical' values, instead of deriving values from empirical confusion matrices. This is because we want these values to reflect only difficulty in *prosodic* perception. However, actual boundary perception can be affected by grammatical structures derived from lexical content (Kuang et al. 2022). For example, in Hegemonic American English, statements often end with rises, and questions with falls (e.g. Bolinger 1999), and this is attested in our Mandarin data too. Though we tell annotators to consider only prosody, not content, they may still be affected by syntax and lexis, e.g. putting a question mark (?) after a syntactic/pragmatic question even though it has falling intonation. Such errors, even if common, need to be counted more heavily than errors caused by acoustic similarity. A possible alternative is to use confusion matrices from expert annotations assumed to *not* contain the syntax-based errors, which we do not pursue in this study because we do not yet have such datasets.

4.3 Results

Similarity scores are shown in Figure 4. As expected, I-based scores are lower than M'_T-based ones, and $S_f > S_f^B$ regardless of the similarity matrix, with S_f values nearing 1. The variation between texts is small within each measure, especially for S_f; there is greater variation in S_f^B.

The κ values are shown in Figure 5, where it is clear that S_f^B-based κ's remain substantially lower than S_f-based ones and I-based than M'_T-based ones. Overall, IAA scores are substantially lower than raw similarity scores, which is expected since they take into account the fact that chance-level similarity can be quite high. There is also less divergence between different measures for IAAs than raw similarities, suggesting that there is less difference as to how much each measure diverges from the chance-level value of that measure.

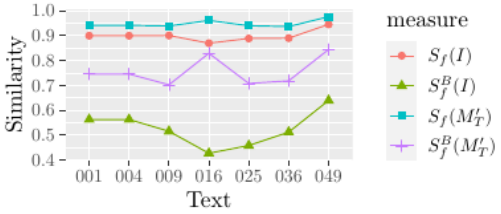

Figure 4: Various similarity metrics applied to texts

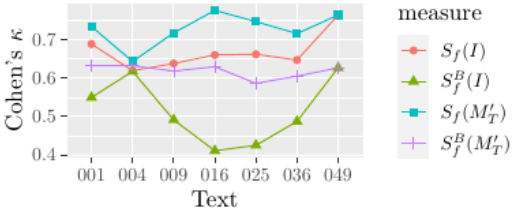

Figure 5: κ values for various similarity metrics.

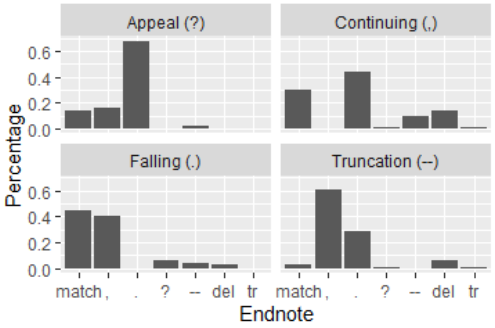

Figure 6: Distribution of operations performed on endnotes. 'Match' means full match, 'del' means deletion, 'tr' means transposition; the rest are substitutions between boundary types. Transposition plus substitution operations are not attested, and thus not shown.

Figure 6 shows the distribution of operations performed on each type of endnote in the annotations. The rate of full matches (i.e. both position and boundary type match) is quite low; falls are matched less than 50% of the time, the rest even less. Yet deletions and especially transpositions are rare, indicating high consistency for boundary *positions*: continuations have the most deletions, and even there the rate is less than 20%. Most of the errors are inconsistencies between boundary types. Truncations often correspond to continuations and sometimes to falls by other annotators. Falls and continuations are often confused for each other, while appeals correspond to falls around 70% of the time.

4.4 Discussion

The distributions of operations explain many of the patterns seen in the similarity score measures. Because most of the operations are substitutions between boundary types, once M'_T is used and correspondences between easily confusable boundary types are thereby downweighed, the similarity score rises drastically compared to I-based similarities. The dramatic disagreement with respect to boundary types may be attributable to a) lexical tone, which complicates perception as listeners must calibrate their perception of final pitch trajectories to the individual lexical tones; and b) the fact that words near IU boundaries, especially final particles, are often spoken very rapidly. Additionally, many appeal endnotes (?) were marked as falling (.) by the other annotator; manual inspection reveals some situations where the pitch contour is clear, but the one of the annotators decided between . vs ? based on syntax or pragmatics instead. Future annotator training will emphasise the importance of ignoring non-prosodic factors and calibrating intonational judgements according to lexical tones.

Notably, even when we consider Cohen's κ, a marked divergence between S_f and S_f^B remains. This is likely partially due to inherent weaknesses with using S_f^B for κ. In calculating chance-level similarity, the simulated annotations will have a comparable number of boundaries to the original annotations, because of how the distribution we simulate from is defined. But random placement of boundaries results in many mismatched boundaries, and hence a larger number of boundaries than actual annotations, which will have much more matches. This artificially inflates $S_f^{B,chance}$ compared to S_f^B, deflating S_f^B-based κ. Thus S_f may be the more suitable choice in κ calculation, and the moderate agreement indicated by S_f-based κ is a better indication of our annotation performance. This matches intuitively with the fact that boundary locations are mostly matched, while agreement on continuations and falls (the most common contours) are fair. The property of S_f^B discussed here may not have been noticed by Fournier (2013), who argued for *B* over *S*, because he focused on cases with full misses (insertion/deletion) and near-misses (captured by transpositions). He did not explore datasets like ours where *substitutions* between boundaries with largely matched positions are the primary operation.

Although we believe the *B*-based denominator is not optimal in this case, we do not claim that *N* is preferable in every scenario. For example, when one's main goal is to compare across texts to evaluate the difficulty of computationally detecting boundaries in each one, normalising with *N* unduly favours texts with sparser boundaries (longer segments). In ongoing work, we applied the measure to a case of evaluating a machine segmenter against different texts to determine the difficulty of segmenting different text types, and preliminary results show that S_f can give misleading results where S_f^B does not. We believe it is best to choose the denominator according to the specific dataset and problem.

5 Conclusion

In this paper, we introduced flexible segmentation similarity S_f, a new edit distance-based measure of segmentation similarity involving multiple mutually exclusive boundaries with fully flexible transposition, substitution, and addition/deletion costs. We justified its properties, presented an algorithm for computation, and extended it to inter-annotator agreement. We applied it to a case of intonation unit segmentation, where we evaluated consistency between manual segmentations and found ways to improve annotator training. We argued that, contrary to Fournier (2013), the number of boundaries is not always the best choice of denominator in calculating segmentation similarity for inter-annotator agreement when there is high agreement on boundary location but low agreement on boundary type. We hope our measure will find other use cases, especially where gradient differences between boundary types are needed.

Acknowledgements

Thanks to Lu Liu, Jack Sun, Sabrina Sun, Danni Wang, Sirui Wang, Haoran Yan and Sunny Zhong for their annotation work, John W Du Bois for valuable guidance throughout the project, Haoran Yan, Olivia Jonokuchi, Laurel Brehm, Simon Todd and UCSB's CEILing group for comments on an earlier draft, Sherry Chien for her involvement early in the project, and Tianrui Gu and Giselle Ramirez for their current work extending the package.

References

Barth-Weingarten, Dagmar. 2016. *Intonation units revisited: cesuras in talk-in-interaction* (Studies in Language and Social Interaction 29). Amsterdam ; Philadelphia: John Benjamins Publishing Company.

Beeferman, Doug, Adam Berger & John D. Lafferty. 1999. Statistical models for text segmentation. *Machine Learning* 34(1–3). 177–210.

Byrt, Ted, Janet Bishop & John B. Carlin. 1993. Bias, prevalence and kappa. *Journal of Clinical Epidemiology* 46(5). 423–429. https://doi.org/10.1016/0895-4356(93)90018-V.

Chui, Kawai & Huei-ling Lai. 2008. The NCCU corpus of spoken Chinese: Mandarin, Hakka, and Southern Min. *Taiwan Journal of Linguistics* 6(2).

DuBois, John W., Schuetze-Coburn, Susanna Cumming & Danae Paolino. 1993. Outline of discourse transcription. (Ed.) Jane A. Edwards & Martin D. Lampert. *Talking data: Transcription and coding in discourse research*. Lawrence Erlbaum Associate, Inc. Publishers.

DuBois, John W. 2020. Discourse Functional Transcription: Conventions. Unpublished manuscript.

DuBois, John W., Terry DuBois, Georgio Klironomos & Brady Moore. 2020. From answer to question: Coherence analysis with Rezonator. In *Proceedings of the 24th Workshop on the Semantics and Pragmatics of Dialogue*.

Fournier, Chris. 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)*, 1702–1712.

Fournier, Chris & Diana Inkpen. 2012. Segmentation Similarity and Agreement. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics*.

Franz, Martin, J. Scott McCarley & Jian-Ming Xu. 2007. User-oriented text segmentation evaluation measure. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 701–702.

Huang, Chu-Ren, Keh-Jiann Chen, Li-Li Chang & Feng-Yi Chen. 1997. Segmentation standard for Chinese natural language processing. In *International Journal of Computational Linguistics & Chinese Language Processing, Volume 2, Number 2, August 1997*, 47–62.

Huang, Chu-Ren, Shu-Kai Hsieh & Keh-Jiann Chen. 2017. *Mandarin Chinese words and parts of speech: A corpus-based study*. Routledge.

Kuang, Jianjing, May Pik Yu Chan & Nari Rhee. 2022. The effects of syntactic and acoustic cues on the perception of prosodic boundaries. *Proc. Speech Prosody 2022* 699–703.

Lin, You-Jing. 2009. *Units in Zhuokeji rGyalrong discourse: Prosody and grammar*. University of California, Santa Barbara.

Lu, Wei & Hwee Tou Ng. 2010. Better punctuation prediction with dynamic conditional random fields. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, 177–186.

Passonneau, Rebecca J. & Diane J. Litman. 1993. Intention-based segmentation: Human reliability and correlation with linguistic cues. *Proceedings of ACL-93*.

Peshkov, Klim & Laurent Prévot. 2014. Segmentation evaluation metrics, a comparison grounded on prosodic and discourse units. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*.

Peshkov, Klim, Laurent Prévot & Roxane Bertrand. 2013. Evaluation of automatic prosodic segmentations. In *Interface Conference 2013 (IDP-2013)*, 95.

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

Qian, Peng, Xipeng Qiu & Xuan-Jing Huang. 2016. A new psychometric-inspired evaluation metric for Chinese word segmentation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* (Volume 1: Long Papers), 2185–2194.

R Core Team. 2022. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Troiani, Giorgia, DuBois, John W. & Gries, Stefan Th. (2023). Testing the perception of Intonation Unit boundaries in naturally occurring conversation. XV International Conference on General Linguistics, University of Madrid Complutense.