A probabilistic error metric for segmentation algorithms *

Doug Beeferman
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
dougb+@cs.cmu.edu

June 3, 1997

1 Introduction

Precision and recall statistics are commonly used in natural language processing to assess the quality of algorithms, and have served the information retrieval community well in particular. A number of tasks in IR do not lend themselves to these measures, however. Segmentation problems, which ask us to divide a stream of linguistic units into coherent segments automatically (where the segments can be phonemes, words, sentences, paragraphs, or documents) are ill-served by precision and recall, because they are less about identifying certain entities in text than they are about detecting transitions between them. In this talk we shall introduce and discuss a new probabilistically-motivated error metric for segmentation tasks that overcomes the weaknesses and complements the strengths of precision and recall.

*Abstract

2 Segmentation algorithms in the real world

Automatic segmentation algorithms are applicable to several kinds of NLP systems. Three examples demonstrate the range of granularity at which they make their decisions:

- A concatenative text-to-speech program might build up a vocabulary of phonetic units from a large sample of audio; extracting these units accurately requires first aligning the training data to the corresponding phonetic sequence. Equivalently, we must identify the positions within the audio signal (between consecutive 10-millisecond frames, say) that most nearly divide consecutive phonemes.

- A more "coarse-grained" segmentation problem is the task of determining the location of sentence boundaries in raw text, a common conditioning step in the markup of electronic documents. Here we're given words, not frames, and must chop the input stream into sentences, not phonemes.

- Coarser still is the problem that motivated this work—topic segmentation. The input to this problem is a sequence of sentences in some domain (say, financial newswire), and the challenge is to determine at which sentences the topic of discourse changes. Such a segmenter can be useful for document summarization and information retrieval. The Informedia project at Carnegie Mellon University, for example, offers a "news-on-demand" system that allows the user to retrieve recent television and radio news clips relevant to specified keywords. Segmentation of the stories, using automatically recognized transcripts and features of the video and audio signals, is a necessary conditioning step to ensure that the user is returned an appropriate amount of context around sentences in the corpus that match his query.

For the purposes of this study we'll define a segmentation algorithm to be a program that takes a sequence of linguistic *units* and outputs a *hypothesis segmentation*, a partitioning of the sequence into *segments* of consecutive units. Equivalently, such a program places boundaries between pairs of consecutive units in the input. We consider only problems in which the segments are exhaustive, nonoverlapping and nonempty, and so for a given sequence of

n units, there are $n-1$ candidate boundary placements, any subset of which an algorithm can select as its output.

For a given input sequence, we assume there to be a single correct *reference segmentation* that defines what "perfect" performance means. The reference segmentation is often directly implied by some richer (but presumably hidden) markup of the data, for example the SGML version of the training data for a document segmentation task. In other cases it can be the work of human judges—deciding subtopic segmentation of conversation transcripts, for instance. Figure 1.

3 Evaluation

In each of the three applications above, segmentation errors can harm system quality. A topic segmenter that makes boundary guesses too conservatively might miss a few reference boundaries, hypothesizing segments that are too large. In the Informedia application this might mean that the user is returned portions of advertisements or story fragments completely unrelated to his query. On the flipside, an over-aggressive algorithm that fragments the input too harshly will truncate relevant stories.

What is the proper balance? We wish to penalize both defects in a principled way; more to the point, we need an evaluation metric that correlates with the utility of a segmenter in its intended application. If the segmenter is a component of a larger system, such as a text-to-speech program, the final measure of its quality may be its impact on existing evaluation measures for that larger system (intelligibility, for example.) However, principles of modular design suggest that the segmenter be optimized independently if possible.

The evaluation of a segmenter independently from its application must then be a function of the *reference segmentation* of its input and the *hypothesis segmentation* produced by the segmenter on it. Choices of this function are considered below.

3.1 Traditional evaluation using precision and recall

Precision and recall are employed by information retrieval researchers to measure the performance of algorithms that *find desirable things*. Informally, *precision* is the fraction of things found that are desirable, and *recall* is the fraction of desirable things that are found. It is always possible to achieve perfection along one of these dimensions at the expense of the other by implementing a trivial algorithm that finds everything or nothing at all, and so the two must be reported together. Where a single figure of merit is desired, weighted combinations such as the *f-measure* (usually $\frac{2 \cdot P \cdot R}{P+R}$) are often used.

In prior work on segmentation, the boundary candidates are seen as the *things*, and the actual boundaries are seen as desirable. An evaluation of a segmentation algorithm's performance using precision and recall tells us what fraction of the hypothesized boundaries are correct and what fraction of actual boundaries were hypothesized. Since the number of atoms in the input may be much larger than the number of segments, predicting the boundaries exactly is a hard problem, and in most cases exactness is neither expected nor absolutely necessary. Yet an algorithm that places a boundary one unit away from the actual boundary every time actually receives *poorer* precision and recall scores than an algorithm which hypothesizes a boundary at every position. (Figure 1.)

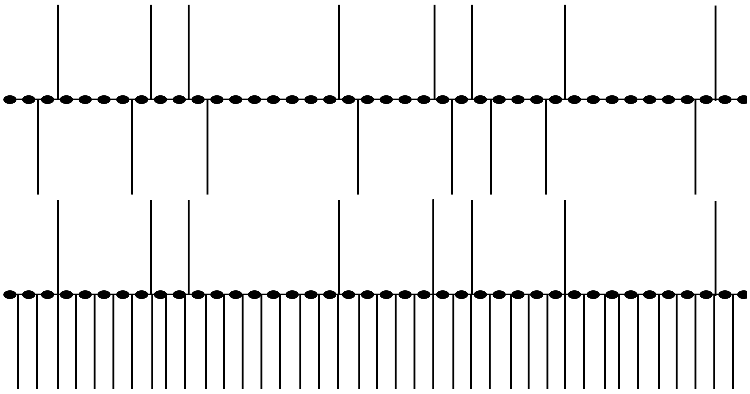

Figure 1: Precision and recall for two hypothetical segmenters on the same 41 units of data. The reference segmentation is represented by the vertical lines above the center, and the hypothesis is represented by the vertical lines below the center. The top algorithm misses the correct boundaries consistently by one unit achieving $P = R = 0$. The bottom algorithm guesses boundaries everywhere, achieving $P = 0.125$ and $R = 1$.

In almost any conceivable application (including the three in the above section), a segmenter that consistently comes close—off by one unit, say—is preferable to one which places boundaries haphazardly. We can patch precision and recall somewhat by treating as "correct" guesses that are within some constant-sized window of units away from a reference boundary, but this revision still suffers from overdiscretizing error, drawing all-or-nothing lines insensitive to gradations of correctness.

3.2 Toward probabilistic evaluation

We shall introduce a new probabilistically-motivated error metric for segmentation tasks that overcomes these problems, summarizing performance in a single number that is more closely related to application performance than precision, recall, or the "f-measure" weighted combination of the two.

An alternate view of segmentation treats *pairs* of units in the corpus as significant, rather than single units in isolation. If we regard the units that comprise each reference segment in a corpus as mutually related, then each pair of units in the corpus can be tagged "related" or "unrelated" with respect to the reference segmentation.

We feel that in most applications the number of pairs that an algorithm labels correctly in this respect is a good measure of its quality. A natural way to reason about developing a segmentation algorithm is therefore to maximize the likelihood that two such units are correctly labeled as being related or being unrelated. Our error metric E_D is simply the *probability that two elements selected from the input are correctly identified as belonging to the same segment or not belonging to the same segment.* More formally, for a corpus n units long,

$$E_D(\text{ref}, \text{hyp}) = \sum_{1 \leq i \leq j \leq n} \frac{1}{n} D(j-i)(\delta_{\text{ref}}(i,j) \overline{\oplus} \delta_{\text{hyp}}(i,j))$$

Here δ_{ref} is an indicator function which is 1 if the two corpus indices specified by its parameters belong in the same document, and 0 otherwise; similarly, δ_{hyp} is 1 if the two indices are hypothesized to belong in the same document, and 0 otherwise. The $\overline{\oplus}$ operator is the `XNOR` function ("both or neither") on its two operands. The parameter D is a *distance probability distribution* over the set of possible distances between units chosen randomly from the corpus. If D is uniform over the length of the text, then the metric represents the probability that any two units drawn from the corpus are correctly identified as being in the same document or not.

In practice letting D be the uniform distribution is unreasonable, since for large corpora most randomly drawn pairs of sentences are in different documents and are correctly identified as such by even the most naive algorithms. We instead adopt a distribution that focuses on small distances. A natural choice for D is the exponential distribution with mean μ, a domain-dependent parameter:

$$D(k) = \frac{e^{-\frac{1}{\mu}\cdot k}}{Z} \tag{1}$$

In the above, Z is a normalization chosen so that D is a probability distribution over the range of distances it can accept. (If we allow arbitrarily large distances, the normalization is $Z = \sum_{k=0}^{\infty} e^{-\frac{1}{\mu}\cdot k} = 1 - e^{-\frac{1}{\mu}}$.)

In this talk, we shall examine how this error metric behaves under various natural and extreme conditions, show how it can be computed efficiently, and motivate the choice of the distance function and its parameter μ. We shall also demonstrate the metric's application to the topic segmentation task described above.