

CS65 Midterm Project: Word Segmentation in English

Matthew Singleton and Stephen St.Vincent

Swarthmore College

{msingle1, sstvinc2}@cs.swarthmore.edu

Abstract

We present here a variety of methods of performing word segmentation. We use mostly statistical methods to segment words and evaluate each segmenter's performance in two measures: precision and recall. A third measure, the f -value, is a way to combine precision and recall into an overall quality measurement.

We find that none of our segmenters are particularly good. This unfortunate result stems largely from the difficulties inherent in obtaining an adequate training corpus. Still, we are able to draw conclusions when comparing different segmentation algorithms to each other. We find that predecessor-based methods are far superior to successor-based methods, suggesting (rather accurately) that English is a suffix-dominated language.

1 Introduction

A common problem in the field of Computational Linguistics is obtaining a corpus of text that is that of segmented at morpheme boundaries. Segmented words are more useful than full word-forms for information retrieval. Various attempts have been made to perform word segmentation in a minimally supervised environment (Hafer and Weiss, 1974) (Harris, 1955) (Harris, 1967). In this paper, we aim to duplicate the methods used in these papers.

2 Methods

2.1 Training Corpus

We began with a provided corpus file, which was a list of all words occurring in the British National Corpus and their corresponding word counts. We then cut this list down by removing all words that matched any of the following criteria:

- words with at least one capital letter
- words with frequency less than 4
- words with length less than 3 characters

We then sorted the corpus alphabetically so that we could process it in one pass.

2.2 Gold Standard

To create our gold standard, we randomly selected 500 words from the training corpus, put them in another file, and removed them from the training corpus to avoid over-fitting. We then hand-segmented these words, removing words that were not English. This left us with 442 hand-segmented words.

2.3 Testing

In total, we implemented 12 different segmentation algorithms, which are various combinations of the following six general algorithms (all but entropy and random are from Harris (1955)):

successor counts For each substring of length n , starting at the beginning of the word, count the number of unique characters occurring at position $n + 1$.

predecessor counts For each substring of length n , starting at the end of the word, count the number of unique characters occurring at position $n - 1$.

successor n+2 For each substring of length n , starting at the beginning of the word, count the number of unique characters occurring at position $n + 2$ for each unique character at position $n + 1$.

negative (forward and backward) For each substring of length n , if the substring is a full word found elsewhere in the training corpus, store a negative value as a flag at position n .

entropy The entropy (H) for each substring (α_i) from the beginning of the word is given by:

$$H\alpha_i = \sum_{i=1}^{26} -\frac{|D\alpha_{ij}|}{|D\alpha_i|} \cdot \log_2 \frac{|D\alpha_{ij}|}{|D\alpha_i|} \quad (1)$$

Where $|D\alpha_i|$ is the number of times each i length substring of α appears in the training corpus, and $|D\alpha_{ij}|$ is the number of times the j th letter of the alphabet was the immediate successor to the substring α_i . $|D\alpha_{ij}|/|D\alpha_i|$ is therefore the probability that the successor to α_i is the j th letter (Hafer and Weiss, 1974).

random We randomly segmented words after after each character with a probability of $1/5$.

Below we describe our 12 segmentation algorithms in the context of the above general algorithms.

Successor counts at peak Segmentations were made whenever a local maximum appeared in the successor counts.

Predecessor counts at peak Segmentations were made whenever a local maximum appeared in the predecessor counts.

Successor N + 2 counts Segmentations were made whenever a local maximum occurred in the $n+2$ counts.

Successor cutoff Segmentations were made whenever the successor count exceeded a cutoff, which was 11.

Predecessor cutoff Segmentations were made whenever the predecessor count exceeded a cutoff, which was 11.

Successor and predecessor cutoff Segmentations were made whenever both the successor and predecessor counts exceeded a cutoff. The cutoff for successor was 3 and for predecessor was 8.

Successor + predecessor cutoff Segmentations were made whenever the sum of the predecessor and successor count exceeded a threshold, which was 22.

Forward negative Segmentations were made wherever the successor flag was set.

Backward negative Segmentations were made wherever the predecessor flag was set.

Entropy Segmentations were made whenever the sum of the successor and predecessor entries exceeded 4.

Random Segmentations were made randomly.

3 Results

Tables 1 and 2 give the statistical output from our segmenting algorithms. The cuts made by the algorithms presented above were compared to the hand-made segmentations in the gold standard. *Precision* is the number of correct segmentations divided by the total number of segmentations made by the algorithm. This measures how likely a given cut is to be correct. *Recall* is the number of correct cuts made divided by the total number of cuts in the gold standard. This, then, measures the fraction of cuts in the gold standard that were also made by the algorithm. The f -value is a statistical measure given by the following equation:

$$f = \frac{2 \cdot p \cdot r}{p + r} \quad (2)$$

where p is the precision and r is the recall. It is difficult to compare two algorithms on the two scales of precision and recall simultaneously. This f -value gives an objective way to determine which algorithm truly had the higher overall performance.

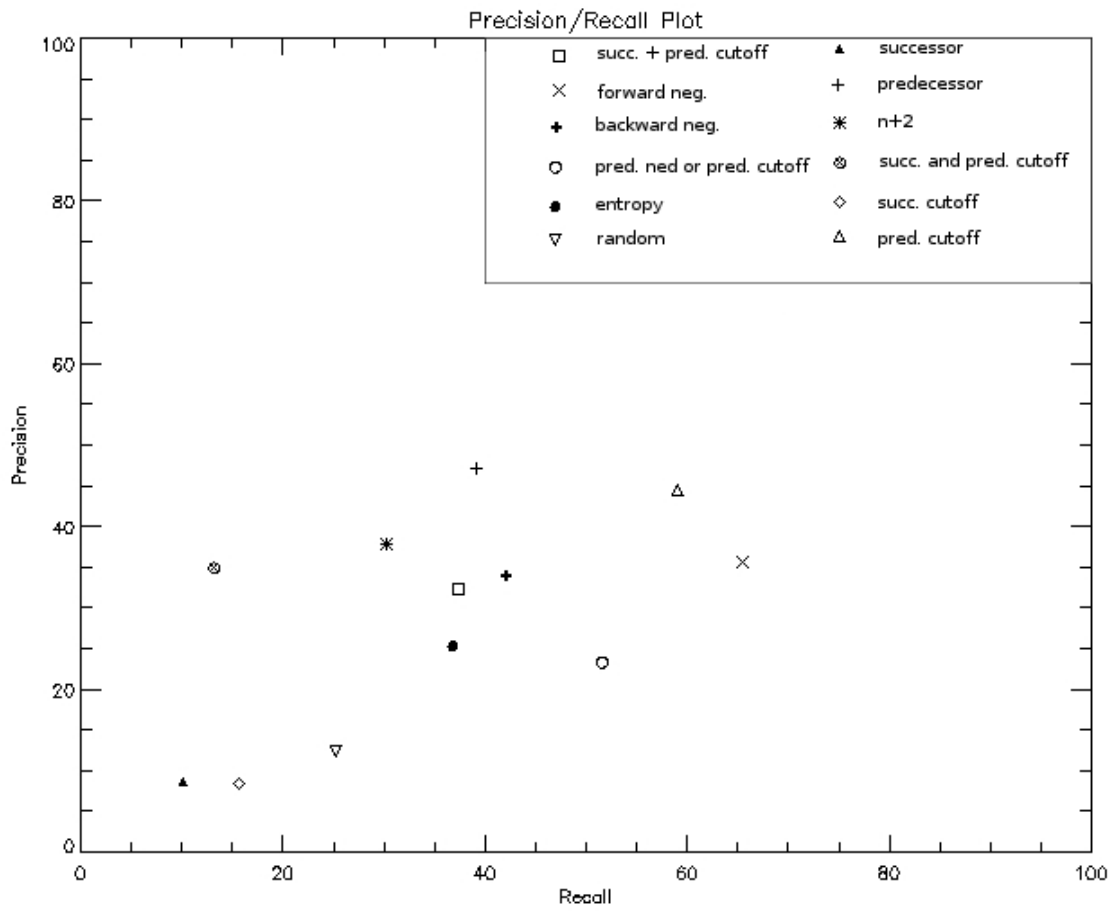


Figure 1: Precision vs. Recall. This plot shows the values of the precision and recall measures for the 12 segmenting algorithms tested. Both axes are measured in percentages.

Figure 1 plots the precision vs. the recall for each algorithm. While it is easy to see that successor did much worse than predecessor cutoff, but it is difficult to compare forward negative and predecessor cutoff. The f -value betrays that predecessor cutoff is superior.

4 Discussion

As our data clearly show, none of the algorithms that we implemented are terribly good at word segmentation. One contributing factor to this lack of performance was the quality of the training corpus. Many words in the corpus were not English words, if they were even words at all. Despite our best filtering efforts, bad words still made it into our training corpus, which significantly hindered our segmenting algorithms.

However, given these data limitations, we can still attempt to draw general conclusions when comparing the algorithms to each other, if not overall. The f -values in table 1 show that the predecessor cutoff algorithm, with an f -value of 0.508, was the best algorithm given the data presented, followed closely by forward negative, with an f -value of 0.459. Another interesting result is that our random segmenter (f -value of 0.177) did better than both successor (0.094) and successor cutoff (0.109) segmenter, suggesting that English is a suffix-dominated language.

Future work could include adjusting the cutoff values for any of the cutoff algorithms and/or the entropy algorithm to achieve a maximum f -value. In addition, (Déjean, 1998) presents an algorithm in which they find the n most-likely suffixes and bootstrap a segmenter from there, which could also be implemented. Finally, (Hafer and Weiss, 1974) present additional combinations of the six general algorithms (section 2.3).

Word segmentation is inherently problematic. Often, two native speakers will disagree on the segmentation of a particular word. In languages such as Chinese and Japanese, the segmentation of the romanized versions of the languages are almost meaningless, as agreement is scarce. Even in English, this task is difficult. Consider the word *muddies*: it could be segmented as *muddi* and *es*, *mudd* and *ies*, or even *mud* and *dies*. It would be more meaningful to try to obtain the true stem of a word (here, *mud*) and its

affixes (here, $-y$ and $-s$).

5 Conclusion

Word segmentation is a difficult task with few objectively correct answers. Our work here demonstrates many of the difficulties inherent to this task. Despite our best efforts, we were unable to even obtain a satisfactory training corpus and gold standard of sufficient length to truly test our algorithms. To do so would be a very time-consuming task. As such, our overall results are greatly skewed downwards, although we believe that our relative results can still be useful, for instance in recognizing that predecessor statistics are more useful than successor statistics.

References

- Hervé Déjean. 1998. Morphemes as necessary concept for structures discovery from untagged corpora. In *Proceedings of the ACL-98 Workshop on New Methods in Language Processing and Computational Natural Language Learning*.
- Margaret A. Hafer and Stephen F. Weiss. 1974. Word segmentation by letter success varieties. *Information Storage and Retrieval*, 10:371–385.
- Zellig Harris. 1955. From phoneme to morpheme. *Language*, 31:190–222.
- Zellig Harris. 1967. Morpheme boundaries within words: Report on a computer test. In *Transformations and Discourse Analysis Papers*. Department of Linguistics, University of Pennsylvania.

Segmenter	Cuts made	Actual cuts	Correct cuts	Precision	Recall	F-value
Successor	526	455	46	0.087	0.101	0.094
Predecessor	378	455	178	0.470	0.391	0.427
Succ. N + 2	364	455	138	0.379	0.303	0.336
Succ. and pred. cutoff	172	455	60	0.348	0.131	0.191
Succ. cutoff	846	455	71	0.083	0.156	0.109
Pred. cutoff	604	455	269	0.445	0.591	0.508
Succ. + pred. cutoff	525	455	170	0.323	0.373	0.346
Forward neg.	820	455	293	0.357	0.643	0.459
Backward neg.	566	455	192	0.339	0.421	0.376
Backward neg or pred cutoff	1010	455	235	0.232	0.516	0.320
Successor entropy	666	455	168	0.252	0.369	0.299
Random	898	455	120	0.133	0.263	0.177

Table 1: Segmenting Statistics. This table shows the raw cut data and the calculated accuracy measures.

Segmenter	# perfect	# words	% perfect
Successor	43	442	0.097
Predecessor	142	442	0.321
Succ. N + 2	123	442	0.278
Succ. and pred. cutoff	97	442	0.219
Succ. cutoff	1	442	0.002
Pred. cutoff	106	442	0.239
Succ. + pred. cutoff	70	442	0.158
Forward neg.	72	442	0.162
Backward neg.	69	442	0.156
Backward neg or pred cutoff	20	442	0.045
Successor entropy	40	442	0.090
Random	29	442	0.065

Table 2: More Segmenting Statistics. This table shows the data for correctly segmenting entire words. Percent perfect is simply number perfect divided by number of words.