CS187: Implementation of Textiling

Styliani Pantela, Dianna Hu, Jonathan Miller, and Kevin Mu

December 1st, 2014

1 Introduction

Description: This implementation is based on the article by Marti A. Hearst, "TextTiling: Segmenting Text into Multi-Paragraph Subtopic Passages".

The algorithm can be broken down into three parts:

1. Tokenization
2. Lexical Score Determination
3. Blocks
4. Vocabulary Introduction
5. Boundary Identification

2 Method

2.1 Tokenizing

We tokenized the article using the following four steps:

1. We turned all text into lowercase and split it into tokens by removing all punctuation except for apostrophes and internal hyphens
2. Group tokens into size w, which represents the pseudo-sentence size.
3. For each token sequence, remove common words that don't provide much information about the content of the text, called "stop words"
4. Reduce each token to its morphological root (e.g. "dancing" -¿ "dance") using nltk's lemmatize function.

2.2 Determining Scores

The paper considers two methods of determining scores for all the potential sequence token gaps. The two methods are block comparison and vocabulary introduction. Block comparison compares adjacent blocks based on the number of words they share and vocabulary

introduction assigns a score based on the number of new words seen in the interval of which the potential gap is a midpoint.

2.2.1 Blocks

This computes a lexical socre for the gap between pairs of blocks. For each gap between adjacent token sequences, we consider (where possible) a set number k of token sequences before the token sequence gap and the same number of token sequences after the token sequence gap. For both of these blocks of k token sequences, we create counts of the number of occurences of each word. Based on this, we use the formula in Hearst to create the actual scores. At a gap i, we have

$$\text{score}(i) = \frac{\sum_t w_{t,b_1} w_{t,b_2}}{\sqrt{\sum_t w_{t,b_1}^2 \sum_t w_{t,b_2}^2}}$$

where t ranges over the tokens in the article, b_1 is the k token sequences before the gap, b_2 is the k token sequences after the gap, and (for example) w_{t,b_1} is the number of occurrences of token t in b_1. High lexical scores (when surrounded by low scores) indicate a topic change.

2.2.2 Vocabulary Introduction

Computes lexical score for the gap between pairs of text blocks. Text blocks contain w adjacent sentences and act as moving windows. Low lexical scores preceded and followed by high lexical scores would mean topic shifts. For this method, w should be the average paragraph length for future implementations

2.3 Boundary Determination

After employing either the block comparison method or the vocabulary introduction method to determine the lexical scores for each of the token sequence gaps, we then determine the resulting boundaries based on the depths of these scores between token sequences. This process involves several steps:

1. Determine the depth score cutoff.
2. Determine the depth score for each possible gap.
3. Compare each gap's depth score to the cutoff.
4. Convert the resulting boundaries to paragraph boundaries.

In step [1], we determine the depth score cutoff based on the formula that Hearst describes, namely the subtraction of the standard deviation of the lexical scores from their mean: $C = \overline{s} - \sigma$ (for the liberal criterion, which here is the default. Using the conservative criterion, $\overline{s} - \sigma/2$, is also available as an option).

In step [2], we determine the depth score for each possible gap with lexical score G by finding the closest local maximum lexical score L to the left of the gap as well as the closest

maximum lexical score R to the right of the gap. The depth score for the gap is then calculated as $DS = (L - G) + (R - G)$.

In step [3], we simply consider a gap to be a boundary if its depth score DS is above the cutoff C.

In step [4], we convert these boundaries, in terms of gap number, to boundaries in terms of paragraph numbers. This conversion is performed by converting the boundaries in terms of gap number to an intermediate form of boundaries in terms of token index, which is finally converted back to boundaries in terms of paragraph numbers (rounding to the nearest paragraph number but disallowing duplicates) based on the results from tokenization.

3 Results

We tested our method on a pubmed article on "Influencing Factors and Applicability of the Viability EMA-qPCR for a Detection and Quantification of Campylobacter Cells from Water Samples" that can be found here: http://dx.plos.org/10.1371/journal.pone.0113812. The method we used to test was precision and recall. Precision is the proportion of the system's outputs that are correct and recall is the proportion of test instances for which the system's output is correct. In this particular case:

$$\text{precision} = \frac{\text{predicted topic breaks} \cap \text{actual topic breaks}}{\text{predicted topic breaks}}$$

$$\text{recall} = \frac{\text{predicted topic breaks} \cap \text{actual topic breaks}}{\text{actual topic breaks}}$$

Here are the results we obtained for the two methods of scoring, namely block and vocabulary introduction:

	block	vocabulary introduction
precision	0.4	0.52
recall	0.18	1.0

4 Future Work

We plan to get a larger corpus of articles on which to test our algorithms. Using this larger corpus, we would like to fine-tune the parameters k and w to produce more accurate results. We also would like to use more advanced statistics (instead of just precision and recall) to measure the accuracy of our implementation.

5 Statement of Work

Kevin wrote the tokenization function and worked to integrate different parts of the code. Jonathan wrote the block-score comparison function. Stella wrote the vocabulary introduction function and worked on preliminary evaluation metrics (precision and recall). Dianna

wrote the boundary detection functions as well as the functions to output the newly split text in human-readable form. We all worked together to merge our code into a working implementation and on the write-up.