Text Swapping Optimization

James Bremner

February 13, 2019

Contents

[1 Introduction 1](#_Toc1121106)

[2 Scope 1](#_Toc1121107)

[3 Input 2](#_Toc1121108)

[Texts 2](#_Toc1121109)

[Swapping Options 2](#_Toc1121110)

[5. Distance Algorithm 3](#_Toc1121111)

[Hamming Distance 3](#_Toc1121112)

[Matching N-grams 3](#_Toc1121113)

[6 Difference Algorithm 4](#_Toc1121114)

[Appendix I: Bid Request 5](#_Toc1121115)

[Appendix II: Examples of Matching N-gram distance calculations 7](#_Toc1121116)

[Appendix III: Example of difference calculation 10](#_Toc1121117)

[Appendix IV: Example of swapping options 12](#_Toc1121118)

# 1 Introduction

An investigation into optimizing an algorithm to generate from a set of input texts a new set of output texts that are maximally different from each other. This work was done in response to freelancer bid request 18572610 ( [Appendix I](#_Appendix_I:_Bid) ).

# 2 Scope

This document describes the text swapping specification and describes possible objective functions with examples of their calculation. Optimizing the objective function awaits final specification of the objective function calculation.

# 3 Input

## Texts

The input is a single file in JSON format with this structure:

File 1 always exactly 1 file

Text 2 to N always at least two texts in file

Section 1 to N always at least one section in each file

Title 0 to N may be zero, one or many titles in each section

Paragraph 1 to N always at least one paragraph in each section

Sentence 1 to N always at least one sentence in each paragraph

Block 0 to N may be zero, one or many blocks in each section

Title 1 always 1 title in each block

Paragraph 1 to N always at least one paragraph in each block

Sentence 1 to N always at least one sentence in each paragraph

All texts will have the same structure: same number of sections, blocks, titles, paragraphs and sentences, all in the same order.

## Swapping Options

Sentences may be swapped in paragraphs. Paragraphs may be swapped sections. Sections may swapped in texts. At run time the user may be specify which swaps are forbidden. In particular, the first and last sentences in paragraphs may be specified as immovable. User can choose to allow all sentences in a paragraph to be swapped, to fix the first sentence in a paragraph or to fix the first and last sentence is a paragraph. The same options apply to paragraphs in sections and to sections in texts.

[Appendix IV](#_Appendix_IV:_Example) shows examples of swapping option selections.

# 5. Distance Algorithm

To optimize the difference between output texts an algorithm is needed to calculate the distance between two texts. Many possibilities exist and have been discussed.

## Hamming Distance

This was mentioned in the bid request. One point is given for each location in the shortest text which has a different content than in the other text. One point is added for each extra location in the longer text. The points are totaled to give the Hamming distance.

This distance measure was rejected because it is easy to maximize by simply changing the content of every location. All such texts have the same distance.

## Matching N-grams

A matching N-gram is defined as a sequence of text elements which appears adjacent and in the same order in both texts.

Matching N-grams are identified in the texts and the texts are converted so that matching n-grams occupy one location.

The distance is calculated as the sum of the following products:

Count of location moves of matching ngrams \* move weight

Count of ngrams present in only one text \* presence weight

[Appendix II](#_Appendix_II:_Examples) shows several examples of the distance calculation in detail.

# 6 Difference Algorithm

The distance algorithm calculates the distance between two texts. However we are required to maximize the differences between all the output texts.

The difference algorithm uses the distance algorithm to calculate the distance between every pair of texts. A statistic is calculated for all the distances. *It remains to be decided* which statistic should be maximized – possibilities are the maximum, the mean, the sum or … IMHO the sum of all distances is the favored statistic.

[Appendix III](#_Appendix_III:_Example) shows a detailed example of the difference calculation for three texts. The output offers maximum distance, mean distance and minimum distance.

Since the number of sentences in corresponding paragraphs can vary in the revised bid request, there are already significant distances between the texts of the input BEFORE any swapping is done. The difference algorithm was run on the sample input file tableau-2500-textes-apres-traitement.json, without any swapping, to give the ‘original’ difference statistics:

Min Distance: 0

Mean Distance: 2.1131

Max Distance: 25

Total Distance: 6,606,080

Note that calculating the distance between every pair of texts is very expensive when there are thousands of texts. The sample input file contains 2500 texts and calculating the difference on a fairly powerful desktop machine requires two minutes. I do not see a way of reducing this expense. If the distances where calculated from just one base text, then all the texts could have a large distance from the base text but all be very similar to each other.

# Appendix I: Bid Request

We have texts composed of sections. In these sections, we sometimes have titles but we always have paragraphs. Inside paragraphs, we have one or several sentences. Sometimes, in one section, we have what we name "blocks". A block is a little group made of one title and one or several paragraphs.

We are developing a tool and we need a mathematician to help us take the best decision.

With this tool, users will import a batch of several hundreds or thousands of texts, gathered in 1 file. All these texts will have the same structure (same number of sections, blocks, titles, paragraphs, all in the same order ). The number of sentences may vary for the corresponding paragraphs in different texts.\*

Users will then define:

1/ Select only a part of the elements:

● If we must use all sections for the output texts or if the tool must use between x and y sections upon the total number of sections that we have in the origin texts.

● Same with blocks inside sections.

● Same with paragraphs inside blocks and sections.

● Same with sentences inside paragraphs.

● If we can sometimes hide the title of a block or if we must "print" it in each and every output text.

2/ Swap some elements:

● If we can swap the sections to get them in a different order in each and every output text.

● Same with blocks inside sections.

● Same with paragraphs inside blocks and sections.

● Same with sentences inside paragraphs.

If we have 37 456 texts in the input file, we must get 37456 texts in the output file.

*\*This is a change from the bid request posted to freelancer, which stated: All these texts will have the same structure (same number of sections, blocks, titles, paragraphs and sentences, all in the same order.*

What we want is to get the most different structures in output between each text. We think this can be achieved by considering that each sentence/paragraph/title/block/section is a distinct element in a sequence. The goal would then be to use the principle of Hamming distance, to get the most different sequences in the output texts. But if you think there's a better way to achieve this goal, we're all ears.

1st question you will have to answer to: is it better to work with smaller sequences (one sequence for one paragraph, then one sequence for one block, then one sequence for one section, then one sequence to select and swap sections) or is it better to work sequences globally, gathering all the elements for 1 text inside 1 longer sequence?

2nd question: are you able to code the algorithm? (it's not mandatory, as long as you can explain the principles to a developer).

# Appendix II: Examples of Matching N-gram distance calculations

The samples are from freelancer chat post dated 11 Feb 2:28pm EST

Enter two outputs: abcd dacb

Outputs with matching n-grams:

'a' 'b' 'c' 'd'

'd' 'a' 'c' 'b'

Distance calculations:

'a' moved 1

'b' moved 2

'c' moved 0

'd' moved 3

Enter two outputs: abcd ba

Outputs with matching n-grams:

'a' 'b' 'c' 'd'

'b' 'a'

Distance calculations:

'a' moved 1

'b' moved 1

'c' present in 1st output only

'd' present in 1st output only

Enter two outputs: abcd bc

Outputs with matching n-grams:

'a' 'bc' 'd'

'bc'

Distance calculations:

'a' present in 1st output only

'bc' moved 1

'd' present in 1st output only

Enter two outputs: dacb ba

Outputs with matching n-grams:

'd' 'a' 'c' 'b'

'b' 'a'

Distance calculations:

'd' present in 1st output only

'a' moved 0

'c' present in 1st output only

'b' moved 3

Enter two outputs: dacb bc

Outputs with matching n-grams:

'd' 'a' 'c' 'b'

'b' 'c'

Distance calculations:

'd' present in 1st output only

'a' present in 1st output only

'c' moved 1

'b' moved 3

Enter two outputs: ba bc

Outputs with matching n-grams:

'b' 'a'

'b' 'c'

Distance calculations:

'b' moved 0

'a' present in 1st output only

'c' present in 2nd output only

The posted examples do not contain any long matching n-grams. Here is an example that shows how long n-grams work

Enter two outputs: abcd dabc

Outputs with matching n-grams:

'abc' 'd'

'd' 'abc'

Distance calculations:

'abc' moved 1

'd' moved 1

# Appendix III: Example of difference calculation

Enter move weight

: 1

Enter presence weight

: 1

Enter outputs space delimited: abcd bcde cdef

Distance calculation abcd V bcde

'a' 'bcd'

'bcd' 'e'

'a' present in 1st output only

'bcd' moved 1

'e' present in 2nd output only

Distance score 3

Distance calculation abcd V cdef

'a' 'b' 'cd'

'cd' 'e' 'f'

'a' present in 1st output only

'b' present in 1st output only

'cd' moved 2

'e' present in 2nd output only

'f' present in 2nd output only

Distance score 6

Distance calculation bcde V cdef

'b' 'cde'

'cde' 'f'

'b' present in 1st output only

'cde' moved 1

'f' present in 2nd output only

Distance score 3

Min Distance: 3

Mean Distance: 4

Max Distance: 6

# Appendix IV: Example of swapping options

Using a text file with two texts, each text containing one section with two paragraphs, each paragraph containing four sentences. Ran a random sentence in paragraph shuffle, three times with swap option ‘all swappable’ and three times with ‘fixed first and last’.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* All swappable

Distance calculation dacbhefg V cabdfgeh

Found sequence 'fg'

'd' 'a' 'c' 'b' 'h' 'e' 'fg'

'c' 'a' 'b' 'd' 'fg' 'e' 'h'

'd' moved 3

'a' moved 0

'c' moved 2

'b' moved 1

'h' moved 2

'e' moved 0

'fg' moved 2

Distance score 10

Distance calculation dcabfheg V acdbhefg

Found sequence 'he'

'd' 'c' 'a' 'b' 'f' 'he' 'g'

'a' 'c' 'd' 'b' 'he' 'f' 'g'

'd' moved 2

'c' moved 0

'a' moved 2

'b' moved 0

'f' moved 1

'he' moved 1

'g' moved 0

Distance score 6

Distance calculation dbcahegf V bacdfegh

Found sequence 'eg'

'd' 'b' 'c' 'a' 'h' 'eg' 'f'

'b' 'a' 'c' 'd' 'f' 'eg' 'h'

'd' moved 3

'b' moved 1

'c' moved 0

'a' moved 2

'h' moved 2

'eg' moved 0

'f' moved 2

Distance score 10

\*\*\*\*\*\*\*\*\* First and last fixed

Distance calculation acbdegfh V acbdefgh

'acbde' 'g' 'f' 'h'

'acbde' 'f' 'g' 'h'

'acbde' moved 0

'g' moved 1

'f' moved 1

'h' moved 0

Distance score 2

Distance calculation abcdefgh V abcdefgh

'abcdefgh'

'abcdefgh'

'abcdefgh' moved 0

Distance score 0

Distance calculation acbdegfh V abcdefgh

'a' 'c' 'b' 'de' 'g' 'f' 'h'

'a' 'b' 'c' 'de' 'f' 'g' 'h'

'a' moved 0

'c' moved 1

'b' moved 1

'de' moved 0

'g' moved 1

'f' moved 1

'h' moved 0

Distance score 4