# 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 and sentences, all in the same order.

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.

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).

Best regards,

Marco.

# Bid

the Hamming distance is the number of positions at which the corresponding contents are different. So: ABCDEF ABCDFE Hamming = 2 ABCDEF FBCDEA Hamming = 2 Are these two pairs equally different? If you think the second pair are much more different, then you need to do more than count the positions that are different, you also need to measure the “distance moved” Since in this case you are permuting the input sequence, rather than replacing sequence elements with a new element from the universe of all possible elements, it is a trivial matter to come up with new sequences all of which have the maximum Hamming distance of N, the sequence length. So I concluded that the Hamming distance is insufficient for your purposes.

I suggest totaling the distance moved for each element between the two sequences, with the requirement that every element moves at least one position.

Algorithm: Compute Hamming distance. If less than sequence length, reject ( zero distance? ) For every element in 1st sequence Compute distance to same element in 2nd sequence Add distance to total

is it better to work with smaller sequences? The answer depends on whether or not you need to preserve the hierarchy e.g. must a sentence remain in the paragraph where it was originally found, or can it move to another paragraph? I suspect that you need to preserve hierarchy, so the answer is you must work with smaller sequences.

Can I code this? Yes.

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.

Your bid request says: “Users will then define … if the tool must use between x and y sections upon the total number of sections that we have in the origin texts.”

I do not understand this requirement. Suppose the input text contain 10 sections, and the user defines the output should contain between 3 and 7 sections. How does the tool decide exactly how many sections, between 3 and 7, to actually include in an output texts?