03 A conceptual approach to editing

It can be very helpful to think conceptually about what we do when we're editing a document: presumably our aim is to enable the document to more effectively communicate the ideas, feelings, etc. of the author to the reader(s), but how do we do that?

At one level you could see it as consisting of three tasks:

• Assessing what the document needs
• Reading through the text
• Making changes to the text

I've deliberately used bullets rather than a numbered list as it's not an order of events. For example, you obviously assess the job before you start reading, but as you actually read the text some aspects of your assessment might change in the light of your growing knowledge and understanding of the script.

Similarly, while a lot of changes can only be made as part of the reading process (i.e. at sentence level), it might be possible to make some changes *before* you start that focused reading. At its simplest, you might for example decide to globally change double spaces into singles.

How/why do you assess the script?

The better you can understand the nature of, and problems in, the script *before* you start reading, the quicker you'll be able to complete the edit (this is also true for proofreading). Therefore, computer tools that offer analysis of the text can be a great time-saver, and can also help you in setting up your stylesheet. (Examples include *HyphenAlyse*, *ProperNounAlyse*, *DocAlyse* and *SpellAlyse*.)

What do you want to change?

This might be an over-simplification but the changes you make can be thought of as (1) to increase consistency (let's include correcting the grammar as 'consistency'), and (2) to improve the meaning/expression.

How/when do you make changes?

Conceptually, I think of changes as being of three types – again with bullets, as this is not a batting order.

• **global** – at it simplest this might be removing all doubles spaces or globally correcting known spelling errors, but it might also include, say, adding a terminal full stop to every figure/table caption. **Focus**: removing inconsistencies

• **selective** – by this I mean searching through the text for each occurrence of 'something', and deciding whether or not to make a change according to the context. The obvious example is Word's own find and replace, where you can click 'Replace' or 'Find Next', but PerfectIt also offers this sort of selective editing, as do a whole range of different macros. **Focus**: removing inconsistencies

• **sentence level** – this is when you actually read through the text and make changes, while you focus on the way the sentence communicates. **Focus**: mainly on meaning, but also removing inconsistencies, and checking that any global changes already made are correct

Which types of change are best?

Changes for meaning can *only* be done at sentence level, but the more inconsistencies you can clear up *before* reading, the better you can concentrate on the meaning (which for me is the really fun bit, anyway). Global changes can be done much more quickly than selective changes, because you're not having to read the context of each, but there's more chance of false positives. For example, if you unwisely try to globally add a full stop to 'etc', you could end up with 'Please fetc.h the ketc.hup!'

Play safe, but build experience

It's up to each individual to decide how 'risk averse' you are: how much you are willing to make global changes to increase your speed, at the risk of making the occasional bad choice and having to revert some of the changes.

My personal advice would be that, in using macros such as FRedit,[1] on the first couple of jobs just use a few obvious, safe changes, but then, as you gain confidence, harvest more adventurous F&Rs from the FRedit library (comes with the FRedit package) and thereby increase your expertise – and your speed.

Play safe – mark the changes

Obviously, you don't need to mark the fact that you've removed all the double spaces, but for most global changes, it's wise to mark them in some way so that, as you read, you will know that a given item has been globally changed.

The other important global 'change' you can make is simply to mark items without actually changing them. The idea is that your analysis will likely show you some aspects of the text that *may* need changing, depending on the context, so those could be globally marked to draw them to your attention as you read.

If changes are being tracked then that is sufficient 'marking', but marking just as an indicator to you as you read (not to the client, later) can be done in a number of ways. Highlighting is perhaps the most obvious, but many editors find it too distracting as they read, and prefer the more subtle technique of font colouring; and you can use different colours, maybe to distinguish between changes made and items to be checked.

Many tools allow you to choose whether to use tracking, highlighting, colouring etc. For example, FRedit, allows this choice of different types of marking to be made for each individual F&R.

Highly INadvisable – in my view

One thing I **never** do is to use any global changes *after* I have finished my read-through of all or part of the text in a file – they might change something in that part of the text that I've already read, and global F&R. Maybe I'm over-cautious, but global F&R can sometimes trip over tracked text.

Edit books per chapter

If you edit books, then for the greatest speed gain it's best to only edit one chapter at a time. You can first run *MultiFileText*, which creates one single file from all the constituent files, and then you can run the analyses on that all-the-text file.

That way you can run your globals on a chapter, read it, and then add to, subtract from or modify any of the items in your global changes list (which can be automatically run by FRedit). Chapter by chapter you will then be making more changes globally, leaving fewer things to be changed at sentence level. Not only does this increase your speed but, with fewer consistency corrections to make, you can concentrate better on each sentence.

[1] FRedit is a scripted global find-and-replace macro: you give it a list of F&Rs, and it does them all globally, one after the other (similar to Jack Lyon's MegaReplacer).