New directions in document formatting: What is text?

Chris Rowley[1)] and John Plaice[2)]

Abstract

What is text? Beyond, that is, material 'tagged' as *CDATA* in the mark-up syntax jargon. Is it just undifferentiated strings of bytes? Or of 'characters'? ... But then, what is 'a character'?

These are the most basic of many important questions whose investigation is timely in light of the rapid global spread of the 'XML paradigm' for documents and networked applications of all types. They lie behind our more radical questioning of current models of text strings in software, asking whether these are adequate even for 'alphabetic scripts' such as those used in Europe, and 'syllabic scripts' used, for example, in Southern Asia, let alone for the 'logographic' or 'ideographic scripts' of Eastern Asia.

Therefore this paper launches a study of 'text in computers', aiming first at a basic understanding of useful abstractions for 'characters' and 'character-strings' and asking whether the same abstractions can be usefully applied to all 'natural languages'? At an implementation level, through this programme we shall determine how best to represent the true nature of 'text strings' in an application-independent way.

This is the important first step on the road to developing the many complex models and algorithms that are required to solve current problems posed by the desire to improve the automated handling of all text, in all contexts and in all its aspects. Our resulting interfaces and implementations will certainly make it feasible for a larger variety of text-handling applications both to achieve far better results and also to communicate usefully with each other.

1 What is the *text* in '*What is text?*'

First we must emphasise that we are not here discussing the structure of the text (in the sense of 'XML-style document structure') but in contrast it is the *text itself*, as represented within computer software, that is our subject matter. Neither are we primarily concerned with low-level representation details for text files (such as Unicode v?.? or Unicode+ or ...) nor with particulars of byte-level encodings (e.g., the 'mime-type'). Rather our concerns lie in those crucial, but sadly neglected, levels between the structure and the encoding of the document:[1]

- What is represented/contained in 'text files' and the 'text fields' of document structures?
- Its atoms, its structure, its transformations.
- Its use by multiple applications.

This is a subject that has never been studied in a coherent way and such essential information about the 'text in a file' is normally non-existent. Indeed, neglect by the software industry has led to the current unsatisfactory paradigm in which, although 'text files' may have some minimal associated information to indicate how to interpret the byte-strings therein, more often this deciphering is left to the Operating System or even to a particular application's intelligence, combined with any assumptions it decides to make about the text. An example of typical current commercial 'best practice advice' for handling multi-lingual text can be found at `http://www.opengroup.org/products/publications/catalog/c616.htm`.

At best, all that is known about a 'text file' will typically be that it is a stream of 'encoded characters' (e.g. it uses code-page cp852 or Unicode v1.0). And even if the file has some internal structure, it probably contains much of its useful information content entirely within such 'text strings' about which little is known, often not even what 'language' they are in!

2 Why ask such a question?

This research had its origins in in our efforts to automate the production of high-quality visual representations of text in a wide variety of scripts

1) Faculty of Mathematics and Computing Open University 1–11 Hawley Crescent London NW5 8NP, UK

2) School of Computer Science and Engineering The University of New South Wales Sydney, NSW 2052 Australia

[1] test
test

and languages [9, 10] and in discussions with other researchers working on related aspects of automated document and text processing [5, 6, 4]. Such input led us gradually to understand that in order to be able to do anything useful with the information content of text it is essential to realise that the useful content of a text string is not all contained in the typical 'character string' that we were trying to deal with.

Further investigation of this phenomenon showed that, although investigation of the nature of 'text' seems strangely to have been eschewed by the experts on document processing (both implementors and researchers), its relevance has been recognised by thinkers in related areas who appear more open to those aspects of text that are independent of any visualisation.

"[We need] to represent digitally the literary forms of connection which could not be represented before."

Ted Nelson

`http://www.xanadu.com.au/ted/XUsurvey/xuDation.html`

"I expect these [ideas] to greatly clarify and speed up the work of prose workers (those who use text without fonts — like authors, lawyers, film-script managers, speech-writers, paralegals)."

Ted Nelson

`http://ted.hyperland.com/TQdox/zifty.d9-TQframer.html`

"Electricity, from the time of the telegraph, whipped language into shape, made it ubiquitous, instant and now, digital."

Derrick de Kerckhove

Text, Context and Hypertext, three conditions of language, three conditions of mind ([3], pages 15-19.)

To these we would add two more:

"It is natural to represent written language in computers as character strings in text files. Indeed, the current trend is to use 'text files' (XML) to represent everything in a computer!"

"That which is potentially never visualised must surely exist independent of its visual form?"

Thus, whilst we started by thinking of text in computers as just an abstraction from the visual form of 'positioned glyphs', we now see text as being important in itself, independent of its visual form. We therefore proceeded to seek a formalisation of the semantics of this abstraction that is useful to all 'text applications', i.e., all those that use this abstraction, if only implicitly.

3 History: characters then and now

All text that we now know from more than about 100 years ago is solely in its visual form and so it can all be described within software using 'glyphs in fonts'; of course, the idea of 'glyph' and 'font' must be interpreted here in a very general and unspecific sense, at least for texts prior to the widespread standardisation of the printing industry. In many ways the biggest change in the accessibility and perception of 'mechanised text' came with the almost simultaneous introduction of 'personal text-visualisation machines' — probably the typewriter was the first — and electro-mechanical communication of text — the electric telegraph. These both date from the mid 19th Century in Europe and North America.

Relatively quickly, as soon as the late 20th Century, these technologies had evolved into their electronic successors which are of course legion, from text messaging to literary archives, and widespread, from farming and fishing villages to communications beyond the limits of our solar system.

Throughout the development of formalised writing, from tally marks through official records to literary manuscripts, on to hand printing, machine printing then automated typesetting, there has been a tendency at each stage to restrict the visual forms available to represent text. For example, the formal style of a particular monastic tradition limits the flexibility allowed to an individual scribe; printing further restricts the available visual forms, and later mechanisation leads to the introduction of a fixed collection of founts, each containing a relatively small and uniform set of glyphs.

The idea of 'the character' as a non-visual artifact emerges very late in this process, and the word was not at all used in this sense until the beginning of the mathematical study of symbol strings in the 1930s. Even then the only uses of the word recorded by the Oxford English Dictionary are for what we here call a'a glyph'. This idea developed from that of a 'simplification of the glyph' that is needed to fit the varied visual forms into a small machine: either a physically small machine like a typewriter; or a digitally small machine such as Morse code for the telegraph or the 5-bit text encoding used on paper tapes in the mid-20th Century (see `http://www.cwi.nl/~dik/english/codes/5tape.htm`). The 'coding solutions' of these technologies had a large impact on the written language of the time, much as 'txt msgs' may be doing nowadays. In particular, the history of the electric telegraph with its language of 'telegraphese'[12], and the important commercial businesses that grew up to support it with code books, operating systems, etc., is extensive and fascinating but largely forgotten.

The rapid changes of the late 20th Century have seen off such size limitations and we now have the basic technology to expand the size and scope of 'text machines' enormously since even the physically small (and shrinking daily) phones and PDAs are 'very big machines'. But maybe we also have the wisdom to know that using this potential can make the task of the software engineer more difficult. However, for whatever reason, applications that fully utilise this vastly increased text-processing power are not being developed, maybe because this remains an ad hoc and potentially difficult, unproductive task as long as that text is just 'strings of characters'. Thus, now that we have the computing power to reverse the trend of simplification that produced and strengthened the rôle of 'the abstract character', it is time to tackle some fundamental questions:

- How can we best use this power to capture the full information content of text?
- Can we retreat from the long process of simplification of glyphs and the emergence of the simplistic notion of the modern 'character'?
- Should we abandon these 'characters' now that they are no longer needed for their original purpose of squeezing text into small machines?

4 Text, scripts and language

Another approach to the investigation of 'the character phenomenon' is to look at whether this concept, as developed to encode only the 'Roman script', makes any sense for other scripts. This study needs to involve various types of expertise, from researchers who study various aspects of scripts to the developers and, most importantly, users of text-based applications in the widest possible variety of languages.

Other papers in this volume suggest strongly that the utility of the 'Western character concept' has not yet been fully evaluated for 'CJK scripts' and we also are aware that that in South Asia there is still much discussion on the meaning and use of 'character-based' encodings of syllabic scripts [7]. So for many (maybe all) of the world's languages, we still need to answer the following questions, even if we accept as useful the Unicode definition of a character:

- What are the 'atomic characters' for various languages?
- Do such 'Unicode characters' exist in real languages?
- Are they useful within text applications in these languages?

Even if we assume that the atoms of the written form of a particular language can usefully be described as 'characters', we still need to ask this question for that language:

- Can written text be usefully represented as simply a 'string of atomic characters'?

It is reasonably clear that this paradigm of the 'atomic character' can lead to a useful, if limited, abstraction of text for 'alphabetic scripts': one in which a simplified form of written text becomes a 'string of characters. But can this idea be productively extended to syllabic scripts?

For logographic scripts there must be radically different models that better represent text; the Unicode approach to such languages, which attempts to represent the text by encoding as 'characters' only a subset of the 'words' in the language, seems bizarre to us. However, we can easily see advantages in adding such word-encodings to the repertoire of possibilities for representing languages such as English where the relationship between the 'spelling', pronunciation and meaning of words is of a similar level of complexity and fluidity.

It is already well established (see [6] on Unicode and language) that 'Unicode strings' will not suffice for any natural language (other than, perhaps, *XMLish* and *Unicodese*!). Something we shall here call a 'language tag' is *always* essential since it is the name of the 'context' in which the 'text string' must *always* be interpreted. Thus this tag is as much part of the text as the words and punctuation within it, they are inseparable. Sadly, although this reality has long been understood and accepted by the Unicode gurus, they still insist that the tag be supplied by XML mark-up and that it is the duty of applications to ensure this inseparability of the tag from the text; we know of no applications (even structure-based editors) that support this concept of 'inalienable tags' so this is not a productive intransigence (see Section 8).

The Unicode standard mandates many properties of individual characters but no properties of the strings that are the only reason for the presence and use of these characters. More generally, there is no standard means for adding to the text, or a part thereof, contextual or structural information. This contextual information, of course, includes the essential 'language tag', but also extends to information about, for example, when the document was written, in what style, in what country; and about how it is to be read, and so on and so forth. It is our general belief that all text should, at least, be supplied with contextual information to an arbitrary level of detail, in a way consistent with the second author's work in intensional versioning [11].

5 Analysis

So what is this thing called 'text'? Or rather, what should it be if it is ever going to be generally useful to all applications?

Possible answers (with commentary) are:

- just strings of bytes?
  yes — at the lowest level;
- just strings of 'encoded characters'?
  often — unfortunately!
- just strings of 'characters'?
  maybe — **but** this leads to the question:
  What are these 'characters'?

Anyone who uses an even slightly advanced text application (especially if they do not use only English text) will understand that intelligible text certainly has to be a lot more than 'character strings', however these are held as byte strings. The complexity (both technical and political) of this statement becomes apparent as soon as one delves into the world of Unicode, the most comprehensive effort so far to reduce the chaos of the current relationships between 'byte strings' and 'character strings'.

But even Unicode itself has been changing rapidly throughout its life; and recently both the pace and the incohesive nature of this expansion seem to have increased considerably. Yet, as Yannis Haralambous and many others [1, 2] continue to point out, the standard still has glaring deficiencies and an almost wholly political extension mechanism.

More pertinent to our current ideas is that the Unicode philosophy brushes aside important questions about 'what characters are' and about their use in conveying information in text. The 'Unicode definition' [14] of a *character* makes them indivisible atoms of meaning but this definition, and its use within the Unicode paradigm, hides many assumptions:

- that such atomicity exists;
- that it is fixed;
- that the 'meaning of a text string' can be derived synthetically from its representation as a string of such atoms.

We are therefore led to the following question (in the title of the next section).

6 What do we do with text?

Perhaps the most general summary of the use of text is to 'Communicate in a natural language'. Whilst we realise that, at least in an aesthetic sense, a text string can be simply *das Ding* (communicating nothing but itself), more often it is intended to hold some information that has an existence independent of the text itself. Expressing ideas in written natural language *inside a computer* is far more than the production of 'character strings' … it is a sophisticated creative process and an important part of the wider-ranging craft of 'information design'.

In our research we do not at all wish to limit what is considered to be a 'natural language'; but we are happy if others do so, for clarity, so long as they allow us to include our particular interests of mathematical and logical expressions and their use in (some) programming languages.

There are many reasons why ‘strings of characters’ are not sufficient for the ‘natural semantics’ of any language; but maybe they are all we have (plus a language tag that gives an application some starting point for manipulating the text). It seems that everyone wants a ‘text file’ to contain just ‘something very similar to what they think written/typeset text looks like’. But why? Because WYSIA(ll)YG is now holy writ? Because text in computers should be no more than a ‘virtual typewriter’?

Of course, visual attributes can be a very important facet of the ‘semantics of text’. The use of a font, such as a bold face or black letter form, can have precise (non-visual) semantics but the modern tendency is to insist that such wholly visual information should not be provided directly but that ‘logical mark-up’ must be used. Despite this, there is still a lot of ‘visual mark-up’ around as is witnessed by the source of this paper!

In light of the above analysis, in answering this section’s primary question we shall start a long way from Unicode’s reductionist, synthetic approach to text. Rather than worrying about its constituents, let us think about what we do with it — in computers, at least. Of course, in this discussion it is important to realise that what is currently done with text in computers is necessarily influenced heavily by the current paradigm of text as simply ‘character strings’. Many things that we believe could be done in computers with natural language information are not currently being developed in applications because the concept of ‘text’ has not yet moved away from the idea of a ‘character string’.

We thus start from the idea that the essence of ‘text in a computer’ is ‘information content’ and its purpose within the computer is to allow many applications to access and interpret that information. We shall hold fast to this operational definition of ‘text’ despite the insistence of some Unicode-zealots that such text should be little more than strings with no inherent structure or meaning. We also realise that it removes from the world of ‘text’ most of the vast collections of material currently being accumulated and stored digitally in vast ‘text (so-called) databases’ and archives but, since in practice these contain solely ‘write-only’ data, this is no great loss!

So what do we do with text in computers? The most obvious answer, and the primary interest of this workshop, is:

- It is written . . . and (maybe!) then read.

These are two quite complex tasks but ironically, with current computing power, neither of these any longer needs an abstract idea of ‘text’ as ‘standardised character strings’, nor the original idea of character as a simplification of glyph.

We could now produce very sophisticated and practical ‘reading+writing machines’ merely by using more sophisticated direct manipulation of huge numbers of ‘glyphs in fonts’. But these ‘Super-Writers’ would not be at all good at doing the other things that people are now doing, or would like to do, with text in computers. This is because a purely visual approach to text will almost completely ignore its information content, leaving all interpretation of the text to the eye and brain of the writer or reader (who may well make very different interpretations).

So here is a very brief list of the things that are either regularly or experimentally done with ‘written text’ in computers. It is written from a user’s viewpoint rather than a technical stance, thus tasks that are computationally very similar may appear in different places and vice versa.

- fill out name-spaces — maybe the most ubiquitous use for today’s software!
- store character-based representations of structured information: numbers, dates (classical database field entries)
- simple matching analysis — spell-check, index, search
- more complex analysis — morphology, grammar-checks, readability checks, pronunciation guides
- archiving and textual study — a large area
- verify the information — not only for those maths/computing languages
- transform the information — e.g. transliteration

- improve the information — e.g. build ontologies
- non-visual presentation of the information — audio, tactile, signing
- produce well-designed visual presentations of its information content

And this list is the starting point for our rethinking of 'What is text?'

7 Research: a first approach

In the light of this analysis we have started a research programme to find useful abstractions to extend and replace 'characters' and 'character strings'. These should be flexible enough to cover all 'natural languages' and rich enough to capture the 'information content' of the text and, finally, they will be application-independent.

Since our current ideas are limited by those languages and scripts with which we are familiar, this project will need input from many people: from linguists and other experts and, more importantly, from users of software in real languages.

Here we make a first proposal for what is needed. It is not to be considered complete, and many aspects will change. In particular, in another article in the same volume [8], we focus on 'characters' and what exactly they might be in a more general setting.

Text should be stored as *contextualised, structured streams*: here a *stream* consists of a string of bytes of known length (called the *basic text*), together with a structured *context*, containing, among other things, information about language, source and presentation. This *context* holds information as to how the bytes are to be interpreted, in a way that can be easily accessed by any application.

This approach is different from the current standard model, using Unicode/XML, in that the *context* is considered to be a core part of the text; thus modifying or removing the context *changes the text*! For the moment, we will limit the structure of such a *context* to being a tree, as was defined and studied in the PhD thesis of Paul Swoboda [13], but we are certain that this restriction to a single hierarchical structure will not be tenable in the future.

The bytes making up the basic text *could* be 'Unicode characters' in some byte-encoding, but we do not impose any such restriction. In addition to using many other possible character encodings, the bytes could be combined, say, to produce indexes into a suitably stored electronic English dictionary, thereby avoiding difficulties such as those of 'US speling vs. UK spelling'.

It is possible that such a system be self-similar, in the sense that the basic text might include information that is itself contextualised. As another example of the flexibility of such a system, at the workshop Yannis Haralambous and Tereza Tranaka [2] put forth the idea that in certain situations characters may themselves be probabilistic entities, because of uncertainties in the reading of the text. Whatever the details of how such concepts might be encoded, they, and many more, can be expressed within our model.

In addition, a wide variety of relations over such *streams* need to be defined. These are needed in order to express and manipulate such structural aspects of text as the atomicity of the components of the basic text, equivalences and orders on *sub-streams* and proximity, for intelligent searching. The *context* then offers information about which of these relations are relevant to a particular text *stream*.

These *contextualised, structured streams* will be transformed by applications to create new *contextualised, structured streams*. Such a transformation might modify some dimensions of the context, or the basic text, or both.

This model is summarised here in tabular form:

- **text**: A stream of bytes of known length with —
    - a **structured context**
      such as:
        - 'language tag'
        - 'source information'
        - 'presentation information'
        - 'reliability information'
        - meta-information on structure
    - and **internal structure relations**
      such as:
        - atomicity
        - equivalences
        - orders

    * proximity

- **text applications**: Transformations of **contextualised, structured streams**

This proposed infrastructure will provide a sound theoretical basis for practical solutions to the complex matching and deduction problems posed by current and future 'text applications' and it will support the communication between such applications of the information content of 'text strings', together with useful contextual information about these strings. Although this is but one of many application areas, this model will provide full support for sophisticated visual presentations of the text.

8 Conclusion: the perils of markup

It should be clear from the current proposal that the authors do not see much utility in the so-called 'higher-order' markup approach to text; we call this the 'standard (XML) model' for attaching linguistic and other information to a text strings.

The main reason for eschewing this approach is that we believe that the *context* must remain at all times — throughout its life — part of the *stream*. The context is an integral part of the text. It is not simply that the context must be attached when text is created. Whenever text is to be transformed, the (possibly transformed) context must remain attached to the new results. Thus the very essence of what is a 'text string' must change from this 'standard model'.

Implementing such a notion of 'stays with' (i.e. 'inalienable tags')is tricky, even using 'structured editors'. In particular, the most important mark-up standards (SGML/XML) do not allow 'marked-up text' in many places where 'natural language text' is needed. And of course the reason that these editors do not work is precisely because the mark-up standards assume that all text is 'just PCDATA' and so all text strings are 'structurally indistinguishable'.

Finally, we wish to note that current XML-style mark-up allows only a single tree, plus arbitrary cross-references. But real-life textual information is structured in ways that are both mathematically well-understood and far richer than such a simplistic model can ever encompass. The claim 'that any kind of internal relationship can be encoded in XML as an IDREF' is not relevant here since, for example, we should be very interested to see a reasonably rich topological structure tractably represented using XML markup.

Thus for this reason alone, a radically new approach to text and other aspects of document representation is needed and we hope that you will now understand why we are taking 'the path less travelled' in our quest and that they will join us in developing our model into a practical but rich environment for multi-lingual computing.

Bibliography

[1] Yannis Haralambous. Unicode et typographie : un amour impossible. *Document numérique* 6(3–4):105–137, 2002.

[2] Yannis Haralambous and Tereza Tranaka. ???. This volume.

[3] M. Herczeg, W. Prinz, H. Oberquelle (Eds). Mensch & Computer 2002: Vom interaktiven Werkzeug zu kooperativen Arbeits- und Lernwelten. B. G. Teubner, 2002.

[4] Roger Kehr. xindy – A Flexible Indexing System. In Proceedings of the EuroTeX'98, pages 223-230, Cahiers Gutenberg, 1998.

[5] Frank Mittelbach and Chris Rowley. The pursuit of quality: How can automated typesetting achieve the highest standards of craft typography? In *Electronic Publishing*, pages 261–273, Cambridge University Press, 1992.

[6] Frank Mittelbach and Chris Rowley Application-independent representation of text for document processing–will Unicode suffice? In Proceedings of the 10th Unicode Consortium Conference: Text, Fonts and Typography, pages 233–246 (`http://www.latex-project.org/papers/unicode5.pdf`), 1996.

[7] John Plaice. Private discussions during a visit to India, 2002.

[8] John Plaice and Chris Rowley. Characters are not simply names: On the semantics of characters. This volume.

[9] John Plaice and Yannis Haralambous. The Omega Typesetting and Document Processing System. http://omega.cse.unsw.edu.au

[10] John Plaice, Yannis Haralambous and Chris Rowley. An extensible approach to high-quality multilingual typesetting. In *RIDE-MLIM* 2003, IEEE Computer Society Press, 2003.

[11] J. Plaice and W. W. Wadge. A new approach to version control. *IEEE-TSE* 19(3):268–276, 1993.

[12] Derek J. Smith. Cryptology and the Electric Telegraph (1853-1865). http://www.smithsrisca.demon.co.uk/crypto-middle.html

[13] P. Swoboda. *Intensional Distributed Programming*. PhD Thesis, UNSW, 2003, forthcoming.

[14] Unicode Home Page. http://www.unicode.org

[15] Extensible Markup Language (XML). http://www.w3c.org/XML

[16] The Extensible Stylesheet Language (XSL). http://www.w3c.org/Style/XSL