12 DEVELOPING TEXT STANDARDS

Susan Hockey
Centre for Electronic Texts in the Humanities
Rutgers and Princeton Universities
New Brunswick, NJ

Text handling has so far played a relatively minor role in museum documentation and information handling systems, but as we move towards the age of multimedia and network-based information systems, the need for standard formats for text is increasingly apparent. Multimedia includes the linking of objects and descriptions of objects to primary and secondary textual sources and the same textual source may be linked to many different objects. Taking the multimedia environment to its logical extension we can envisage the distributed multimedia system where information is held in different locations and is accessed over the network and used for different purposes, whilst still being linked together. Some agreed commonality of format for that information, whether it is text, image or sound, is therefore essential for the usefulness, development and longevity of the data. This paper discusses the particular nature of text and methods of handling it, and examines current thinking in text standards. The term 'electronic text' will be used to mean anything textual in electronic form.

Why text in electronic form?

It is useful to begin with a survey of the uses of electronic texts, or rather the reasons why textual material is put into electronic form. The most obvious of these is of course the preparation of printed copy by wordprocessing, desktop publishing, text formatting or typesetting. Even today, in most cases, the electronic text is considered as a means to an end, the end being an elegantly printed catalogue or an annotated bibliography and the like. The copy is usually prepared by a wordprocessing program followed by desktop publishing, and often much effort is expended in creating a printed page which is aesthetically pleasing, contains all the right characters and has footnotes and other annotations exactly where they are wanted. Each wordprocessing program has its own format of text and its own method of handling footnotes, extra characters and the like. Text formatters and typesetting programs provide similar facilities by commands embedded in the text. Each of these again has its own text format. Anyone who has edited any collection of material is used to the problems of bringing together incompatible wordprocessor formats, but even if solutions are available for this problem, we need also to consider how these documents may be kept for future updating or how they may be merged with material from other sources at some later date.

Electronic text is also created for structured databases and retrieval systems, where the primary function is to search the texts for instances of specific words or terms. Software for interactive full-text retrieval is widely used in many applications. Typical programs include BASIS, BRS-Search, and STATUS. Each of these includes an indexing module which must be used before any text can be searched. Searches are performed on the index and so the response to any search is only as good as the index. These programs provide some choices in building an index, e.g. in stop words etc, but hyphens, apostrophes, foreign words, Roman numerals and the like often cause problems since decisions on how to handle these must be made at the indexing stage. Variant spellings of the same word are also usually indexed separately, which means that all forms have to be given as a search term. Further questions are raised by ambiguous punctuation, for example where a full stop can indicate the end of a sentence, a decimal point or an abbreviation.

These programs tend to respond first by indicating how many documents are hits, but in some types of data it is not clear what constitutes a document. It could be a complete text, or one page or a section of a document. There is also often an assumption that every text is either a bibliography with a recognised structure (author, title etc) or consists of paragraphs each of which consists of sentences. This is not the case for inscriptions, papyri, poetry and many other types of information. A more flexible and extensible data structure is needed.

Text is often also stored in a relational database. Even with variable length fields, a pre-defined structure such as this often becomes limiting as more data is acquired. The sorting and indexing facilities in relational databases are simple and often not adequate to handle historical or multi-lingual material. Very few of these packages acknowledge a date before 1900 AD or a non-decimal currency or a name which does not follow Western European conventions. Designing a relational database requires a good deal of prior knowledge of the relationships between the items of information. This is often not the case with text (and with other historical material) when a database is being created for the purposes of establishing those relationships. It can therefore be argued that both conventional interactive retrieval systems and structured databases are constricting. The data has to be made to fit the software when what is needed is software to fit the data.

Hypertext and multimedia do provide a better means of modelling data in that, with good hypertext software, there are no restrictions in defining the relationships between the elements in the data. Yet many hypertext systems do not, in my view, pay enough attention to the textual part. Effort is concentrated on images, sound and video and the text is seen as subsidiary material. Yet, more often than not, it is the text which provides the link to other related scholarship which is not included in the particular multimedia system. Tools for manipulating the text are not well provided and where they are provided, they tend to concentrate on changing the display of the text (font change, sizes etc) rather than helping the user understand it.

Markup and the multi-purpose electronic text

To be of any use at all, an electronic text needs markup or encoding embedded within it. A text without any markup cannot be manipulated effectively by computer software. Markup is used to specify areas of text which are to searched by a retrieval system (e.g. all titles, or all the text written by author Smith). It also provides further information about material which is retrieved, for example the identifiers of all the documents which are hits or the whether the item retrieved is in a title or part of a subject index. Wordprocessing and typesetting programs use typographic markup, for example italic to identify titles, or bold face or a larger size of type to identify headings. However italic may also be used for other features, for example foreign words in a text or for emphasised words. If the titles within that same text are to be searched, it becomes impossible to distinguish them from foreign or emphasised words. The history of encoding and markup shows the development of many different schemes. For the kinds of text analysis which scholars in the humanities have been performing for years, e.g. concordances and text retrieval, schemes were developed to denote the canonical reference structure of scholarly texts. The MARC record is an encoding scheme for the description of bibliographic and related information. For formatting and printing, a parallel group of markup schemes was developed, most notably that used by the typesetting program TEX, as well as Scribe, TROFF and some typesetter-specific schemes. The internal format used by some wordprocessing programs is not unlike these markup schemes. The inevitable result of this plethora of encoding schemes has been time wasted on conversion, lack of adequate documentation, texts unusable for any purpose other than that for which they were originally created, and the inability to extend the coding in a text. Creating an electronic text is not a trivial task. It therefore makes sense to create text which can be interchanged between different applications and be used for new, and as yet undiscovered purposes, without the need for constant reformatting and reorganisation of the data.

The incompatibility of markup systems has led to the development of generalised markup in the form of the Standard Generalised Markup Language (SGML) which now provides a means of creating such a text. SGML became an international standard in 1986. It is not itself an encoding scheme. It is a meta-language in which markup codes or tags can be defined. The principle of SGML is 'descriptive' not 'prescriptive', that is, it provides a means of describing or marking the components of a text. The processes which are to be performed on the text are functions of whatever computer program operates on the text. Typical components of a text may be title, author, paragraph, sentence, document number, or they may be features such as quotations, lists, names, addresses, dates etc. The components are marked by encoding tags within the texts and what is called an 'SGML application' provides the set of tags for one application area.

At one level, SGML considers a text to be composed simply of streams of symbols, which are known as 'entities'. An entity is any named bit of text and an entity definition associates a name with a bit of text. One use for entities is to encode characters which are not on the keyboard. For example the entity reference &beta; could be used for the Greek letter beta, or &alef; for the Hebrew letter alef. Although, this may seem a clumsy way of encoding non-standard characters, it is needed for transmission across all networks and a computer program can convert from other machine-specific character formats to entity references. A second use is for expanding abbreviations, for example &TEI; for Text Encoding Initiative.

At a higher level a text is composed of objects of various kinds, which are known as 'elements'. These identify the various components of a text, which are whatever the compiler of the text chooses to encode. Each element is marked by a start and end tag. For example:

... the novel <title>Pride and Prejudice</title> is associated with ...

Here the title of the novel is tagged as a title. Angle brackets delimit the tags with the end tag beginning with </.

Attributes may be associated with elements to give further information about the element. For example, for the tag *<chapter>*,

<chapter n=3> ... text of chapter ... *</chapter>*

to give the number of the chapter, or for the tag *<name>*

type - type of name

normal - normalised form

<name type=personal normal='SmithJ'>Jack Smyth</name>

This would enable an index of personal names to be made in which Jack Smyth would be listed under SmithJ. Attributes can also be used extensively for cross-references, which are some choices in building an index, e.g. in stop words etc, but hyphens, apostrophes, foreign words, Roman numerals and the like often cause problems since decisions on how to handle these must be made at the indexing stage. Variant spellings of the same word are also usually indexed separately, which means that all forms have to be given as a search term. Further questions are raised by ambiguous punctuation, for example where a full stop can indicate the end of a sentence, a decimal point or an abbreviation.

These programs tend to respond first by indicating how many documents are hits, but in some types of data it is not clear what constitutes a document. It could be a complete text, or one page or a section of a document. There is also often an assumption that every text is either a bibliography with a recognised structure (author, title etc) or consists of paragraphs each of which consists of sentences. This is not the case for inscriptions, papyri, poetry and many other types of information. A more flexible and extensible data structure is needed.

Text is often also stored in a relational database. Even with variable length fields, a pre-defined structure such as this often becomes limiting as more data is acquired. The sorting and indexing facilities in relational databases are simple and often not adequate to handle historical or multi-lingual material. Very few of these packages acknowledge a date before 1900 AD or a non-decimal currency or a name which does not follow Western European conventions. Designing a relational database requires a good deal of prior knowledge of the relationships between the items of information. This is often not the case with text (and with other historical material) when a database is being created for the purposes of establishing those relationships. It can therefore be argued that both conventional interactive retrieval systems and structured databases are constricting. The data has to be made to fit the software when what is needed is software to fit the data.

Hypertext and multimedia do provide a better means of modelling data in that, with good hypertext software, there are no restrictions in defining the relationships between the elements in the data. Yet many hypertext systems do not, in my view, pay enough attention to the textual part. Effort is concentrated on images, sound and video and the text is seen as subsidiary material. Yet, more often than not, it is the text which provides the link to other related scholarship which is not included in the particular multimedia system. Tools for manipulating the text are not well provided and where they are provided, they tend to concentrate on changing the display of the text (font change, sizes etc) rather than helping the user understand it.

Markup and the multi-purpose electronic text

To be of any use at all, an electronic text needs markup or encoding embedded within it. A text without any markup cannot be manipulated effectively by computer software. Markup is used to specify areas of text which are to searched by a retrieval system (e.g. all titles, or all the text written by author Smith). It also provides further information about material which is retrieved, for example the identifiers of all the documents which are hits or the whether the item retrieved is in a title or part of a subject index. Wordprocessing and typesetting programs use typographic markup, for example italic to identify titles, or bold face or a larger size of type to identify headings. However italic may also be used for other features, for example foreign words in a text or for emphasised words. If the titles within that same text are to be searched, it becomes impossible to distinguish them from foreign or emphasised words. The history of encoding and markup shows the development of many different schemes. For the kinds of text analysis which scholars in the humanities have been performing for years, e.g. concordances and text retrieval, schemes were developed to denote the canonical reference structure of scholarly texts. The MARC record is an encoding scheme for the description of bibliographic and related resolved into concrete references only when the text is processed. In simple terms this is how SGML handles hypertextual information.

The SGML Document Type Definition (DTD) defines the elements which make up a document, giving relationships between them. This is called the content model. A very simple example could be a play which has a content model of acts which are composed of scenes which are in turn composed of speeches. The DTD is used by an SGML 'parser' to validate a document, by checking that it conforms to the model which has been defined.

One important characteristic of SGML is that tags which are SGML-conformant can be used to describe non-SGML material. This means that images or sound or anything else which is not text can be embedded in an SGML-encoded text or that an SGML-encoded text can contain pointers to non-textual material stored elsewhere. SGML therefore provides an extremely flexible shell for multimedia information as well as text alone. The Perseus project (Mylonas 1992) is one excellent example of how SGML-encoded text can be built into a multimedia system.

The Text Encoding Initiative

The requirements of the museum community and the work developed by the Text Encoding Initiative (TEI) project overlap in several areas. The TEI is a major international project to develop an SGML tag set for use by the humanities and language industries. It is sponsored by the Association for Computers and the Humanities, the Association for Computational Linguistics and the Association for Literary and Linguistic Computing. Following a planning meeting in November 1987, funding was provided by the National Endowment for the Humanities, the Commission of the European Communities and the Andrew W. Mellon Foundation. The TEI published the first draft of its guidelines in July 1990 and this has been substantially revised and expanded. The second draft is initially being published in fascicles on the TEI's electronic discussion list TEI-L@UICVM. DTDs are also available from the TEI-L fileserver.

For the first draft of the guidelines the TEI set up four Working Committees. The Committee on Text Documentation, with expertise in librarianship and archive management, was charged with the problems of labelling a text with in-file encoding of cataloguing information about the electronic text itself, its source and the relationship between the two. The Committee on Text Representation addressed issues concerning character sets, the logical structure of a text and the physical features represented in the source material. The Committee on Text Analysis and Interpretation devised ways of incorporating analytic and interpretive information in a text which can include more than one interpretation on the same component of a text. The Committee on Syntax and Metalanguage Issues worked on producing recommendations on how SGML should best be used.

For the second draft of the guidelines, a number of small work groups addressed specific areas in more detail. These included character sets, text criticism, hypermedia, formulae and tables, language corpora, physical description, verse, performance texts, literary prose, linguistic analysis, spoken texts, historical studies, dictionaries, computational lexica and terminological data.

The TEI proposals give guidance both on what features to encode and how to encode them. Although the TEI is proposing between 250 and 300 different tags, very few indeed are absolutely required. The basic philosophy is 'if you want to encode this feature, do it this way'. Sufficient information is provided for the DTDs to be extended by users. TEI conformant texts consist of a TEI header, which provides the documentation or labelling, followed by the text.

The TEI header is believed to be the first systematic attempt to provide in-file documentation of an electronic text. An outline of the header, showing the four major sections, follows:

<TeiHeader>

<fileDesc> ... </fileDesc>

<encodingDesc> ... </encodingDesc>

<profileDesc> ... </profileDesc>

<revisionDesc ... <revisionDesc>

</TeiHeader>

A key objective in the design of the TEI header was to ensure that some elements could map directly on to fields in a MARC record so that catalogue records can be created automatically from the header. A further advantage of using SGML to document an electronic text is that the same syntax is used for the documentation and for the text itself. It can be processed by the same software and maintained in the same way.

The future of SGML

The future for SGML looks now very promising. It is gaining acceptance in many circles. It offers a sound intellectual basis for encoding text which can be used for many purposes and thus provides what Rubinsky (1993) so aptly describes as the 'underground tunnels', which are needed to build the electronic information age. The development of SGML software now must catch up with the applications. SGML-aware software is needed at several stages. There are various software tools to aid conversion from non-SGML encoded text. An SGML parser validates the tagging to ensure that it is processable. There is still a need for good SGML-based browsing software. Dynatext from Electronic Book Technologies of Rhode Island is the only true SGML browser/searcher that I am aware of. PAT from Open Text Corporation of Waterloo searches text which contains SGML-like tags but does not require them to conform to a DTD. Other efforts to create SGML browsers are based on SQL and are developed from the relational database model used by SQL. The choice of software tools will soon grow. It makes sense now for anyone creating electronic text to use SGML if they want that text to last.