Views of Text

Frank Wm. Tompa
Department of Computer Science
University of Waterloo
Waterloo, ON, Canada
N2L 3G1

fwtompa@uwaterloo.ca
http://daisy.uwaterloo.ca/~fwtompa

November 1997

Abstract

Text databases are becoming increasingly important in business applications. However the diverse nature of text is not widely understood, nor appreciated. Some properties of simple document models and text algebras are presented, and these are briefly related to conventional relational systems.

1 Text as a string

The simplest view of text is as a string of characters. Such a string has the property of linear order, but imposes no other constraints. Each character in the text can be identified by its position within the character sequence (i.e., the offset from the start of the text), and substrings can be identified by the positions of their first and last characters. Text matching, then, requires finding substrings that satisfy given predicates, whether for exact match or approximate match, and returning the character positions of the start and end of the matches. Proximity matching can be applied by comparing the character offsets of substrings' endpoints as appropriate.

Although extremely simple, the flexibility of this view allows for arbitrary character sets and arbitrary character encodings. Thus it is equally suited to texts in all languages and genres. This view also allows text operators to be applied to arbitrary sequences of tokens drawn from non-literary alphabets, such as molecular structures considered as protein sequences[9].

Many applications require additional structure to be imposed on texts. Early text editors, for example, viewed texts as sequences of bounded strings, each such string being a *line*. Under this model, text operators were constrained to deal either with a string within one line (e.g., for matching) or with whole lines of text (e.g., for retrieval). Common file operators, such as Unix's **grep** and **diff** take this view of text. Such an approach, however, imposes a dependence on arbitrary line boundaries. Consider, for example, finding all instances of the phrase "data structure" in a text: the view of text as a sequence of lines makes this task more difficult than necessary since we need to account for instances of the phrase that cross line boundaries. Similarly, using **diff** to compare two versions of a formatted essay is often frustrating since white space and new lines may have been inserted arbitrarily.

Alternatively, natural language text can be considered to be a sequence of *words*, each word being a short string of characters. This view of text has many similarities to the simple string model, replacing the alphabet of characters by the set of all possible "word tokens." Substrings are now identified by the positions of their first and last words, where positions are identified by the number of words from the start of the text. Proximity is similarly measured in terms of word offsets.

Viewing text as a sequence of words, however, is troublesome due to problems of word segmentation and of variant forms. English text uses white space to mark word boundaries in most cases, but identifying word boundaries must account for punctuation as well. Furthermore, hyphenated words, such as "text-intensive" are sometimes better treated as one word

International Symposium on Digital Media Information Base (DMIB'97), Nara, Japan, November 1997

and sometimes as two. Variant forms of words, such as plurals and alternative tenses of verbs, also frustrate simple text matching, and force applications to rely heavily on approximate matching or matching words using regular expressions. These problems are even more pronounced in languages such as German or Finnish, where long compound words often appear and yet applications wish to identify these occurrences by querying on their constituent words. In languages such as Japanese, where white space is not used to mark word boundaries, the problems of word segmentation have caused Japanese information retrieval systems to rely primarily on the simpler string model of text[25].

An interesting model of text was used as the basis of the PAT language and system[12, 21]. A text can be considered to be a collection of overlapping semi-infinite strings of characters:

```
A semi-infinite string starts at ...
  semi-infinite string starts at ...
       infinite string starts at ...
                string starts at ...
                       starts at ...
                              at ...
```

String starts can be chosen to be at any (or every) character position, and in this example they are chosen to be at the starts of all words. Each semi-infinite string is associated with a character offset in the underlying simple string of characters. Text matching is the problem of finding all semi-infinite strings that begin with a string that matches a given pattern; it typically returns the character offsets corresponding to the string starts. A substring in the text is identified by a pair of semi-infinite strings, corresponding to the start and end of the substring.

PAT's model of text is a good compromise between the simple string model and the word model. Like the string model, it can be applied to arbitrary character sets and arbitrary character representations, thus making it suitable for non-linguistic applications[13]. When the text represents natural language, however, word boundaries can be used to identify suitable starting positions for the collection of semi-infinite strings. Nevertheless, because words are not themselves fundamental units in the model, applications can operate even with incomplete word segmentation.

2 Tagged text

Text is usually a rich combination of content and form. The latter is typically recorded using some sort of *markup*[6], which is represented by special strings of characters, or *tags*, embedded in the text. Tags can be unusual character codes that do not encode valid content characters (e.g., non-Ascii codes), or they can be reserved character sequences that do not represent valid content "words." In the latter case, a tag is usually distinguished from a text's content by beginning with a reserved character (or short sequence of characters) that may be disallowed in the pure text (e.g., the single character < for HTML and the character pair consisting of newline and dot for troff).

When markup is interleaved with content, it can be subjected to the same operations as the rest of the text. From this arises PAT's and **grep**'s model of tagged text in which tags are merely embedded strings. For example, one can find all top-level headings starting with the phrase "Digital Library" by searching for the string `<H1>Digital Library`.

A simple markup system allows individual tags to be placed arbitrarily in the text. In processing the text sequentially from the beginning (e.g., to format it), each tag behaves as a signal to change the state of the processing engine[20]. As discussed in the next section, tags can also be used in more constrained settings to encode hierarchical and other structured texts.

3 Structured text

The simplest structured model characterizes text as a collection of *documents*, each of which is a simple character string as described in Section 1. In this model, a document plays the role of a record in a conventional database: it is both the unit of retrieval and the unit of update. Thus searches for matching strings behave as document queries that produce, as results, sets of documents containing those strings rather than references directly to the strings themselves. The boolean logic common to database systems extends to such document systems, in which "postings lists" for words or phrases correspond to conventional "inverted lists" [8, 17]. More sophisticated document systems, in which retrieval is based on document ranking instead of exact match retrieval, also rely on this model of text[8, 23].

A slightly more sophisticated model of text subdivides documents into named fields, for the purposes of formatting and for restricting searches. Consider, for example, a bibliographic record in a `bib` file[16], which can include text fields named "author," "title," "publisher," etc. This corresponds exactly to the use of named fields in conventional database records: the document remains the unit of update and retrieval, but queries are phrased in terms of boolean expressions involving attribute-value pairs.

Generalizing from this, fields can have nested subfields. This is the model behind SGML, the standard language for describing highly structured text[10] and XML, an emerging simplified standard[4]. Text regions within a document are marked by pairs of start- and end-tags (see Section 2) that delimit the extent of each region and classify regions by type. Documents that have similar structure share a *document type definition*, which uses a grammar to describe the set of possible region types (i.e., the valid tags) and the nesting and ordering constraints on regions within those documents.

There is a strong parallel between grammars for documents and schemas for conventional database records[11]: grammars are constraints on sets of documents that help to record the structure, determine the validity of newly-added data, and guide query optimizers. Whereas the first of these functions is valuable in document collections, experience has shown that the second role is far less useful in managing document collections than in conventional databases. In practice, the set of valid forms of document evolves very rapidly — almost at the pace of acquiring new documents. Thus database managers for document collections must support many variants of document type definitions for sets of closely related documents. The role of grammars as constraints to guide query optimizers is therefore quite limited.

When text structure is defined by a context-free grammar, documents can be modelled and manipulated as trees[1, 18, 22]. This provides the basis for an algebra in which common needs are conveniently expressed.

Interestingly, the region model of text can also be effective when there are very few constraints on the relative placement of regions in text. In PAT, named collections of regions can be defined dynamically as an aid in specifying queries[21]. The only constraint on regions is that two regions within one named collection (e.g., two "paragraphs" or two "titles") may not overlap each other; regions having different types can co-exist without constraints. Thus, in Shakespeare's plays, for example, some speeches can contain several lines, some lines can contain several speeches, and some speeches can start in the middle of one line and end in the middle of another one. The MultiText model of regions is similar, except that regions of one type may overlap as well — as long as they are not self-nested[5].

4 Applying relational technology to document management

By considering documents as entities, conventional database technology can be adopted to manage a document collection. In a relational database system, each document can be stored as a single value for one attribute, as a single tuple with several attributes, or as a collection of related tuples in one or more relations. These alternatives correspond to the structured text models that consider a document as a single string, a fielded record, or a nested structure (but stored in a normalized form).

Standard support for storing documents in relations, however, is unsatisfactory, as most systems cannot help manage the data as rich structures. Consider, for example, the bibliography for the article entitled "Canada, History of" in *The New Encyclopædia Britannica — Macropædia* Vol. 3, p. 751:

> W.L. MORTON, *The Kingdom of Canada*, 2nd ed. (1969), is the fullest one-volume history and the most traditional.... To understand the place of the colonies that became Canada in the British Empire, the following are most useful: H.A. INNIS, *The Fur Trade in Canada*, 2nd ed. (1956), and *The Code Fisheries*, rev. ed. (1954);... The following works both introduce and analyze the development of the remaining British colonies to self-governing communities and their union in confederation. W.S. MACNUTT combines in a single narrative the histories of the Atlantic provinces in *The Atlantic Provinces, the Emergence of Colonial Society, 1712-1857* (1965). FERNAND OUELLET in his *Histoire économique et sociale du Québec, 1760–1850* (1966; Eng. trans. in prep.), applies with great success the demographic method of French historiography to the little known domestic development of that province....

If such a text were stored in one field, encoded as a VARCHAR or as a BLOB, substructures would not be recognized by the database system itself; rather they would be left to be identified and manipulated purely within application code. In general, whole texts, or significant portions of texts, must be retrieved as entire objects when selection criteria are met, even if only specific subfields are required. If instead a database designer chooses to use a nested, but normalized, approach, the text must be fragmented in a way that is often unnatural. Furthermore, applications must later reassemble document components from scattered fragments through extensive join operations that reconstruct their nesting structure and ordering[19].

We propose instead that structured text be recognized as a datatype in its own right[2, 3]. Operators to extract and manipulate subtexts can then be defined to provide application programs with effective means to access whole texts or text fragments as needed. Using this approach, a document can be stored as a single text value or it can be fragmented to the extent that is appropriate to applications' needs. For example, a database designer may choose to store the above bibliography as a single unit or to store each bibliographic entry as a separate text in its own tuple. Furthermore, given appropriate operators to manipulate texts within the relational model, views of texts can be defined as relations against which further queries can be posed.

In our proposal, a `text` value is created by applying a parser to a string. The result is a stream of word tokens over which is superimposed one or more tree structures, depending on the parser. Subtree matching operators[15] and fulltext string matching operators[14] can then be used together to identify subtexts of interest, and these can be extracted as tuples that preserve the interrelationships among the fragments.

Consider again the above bibliography and imagine that it is stored as a single text value. The parser may well identify a tree in which each entry corresponds to one node, under which there are nodes corresponding to author, title, edition, and date. The qualifier `<entry>#[<title>{Canada}]` matches all entries in which the title field contains the word Canada and `<entry>[<author>#&<title>{Canada}&<date>#]` instead identifies author-date pairs for all such entries (as indicated by the placement of the two # symbols). The operator *extract_subtexts* can be used in conjunction with the former pattern to extract a relation in which each matched entry is stored as a separate text, or in conjunction with the latter pattern, to extract a relation in which each tuple stores the author in one field and the corresponding date in another field.

Consider now a database holding the collected works of Shakespeare. The database designer can choose to store the complete text as a one-row, one-column relation, over which one can use text extraction operators to define several views. For example, one view might be a relation in which each work is represented in its own tuple, with fields to hold metadata extracted from the text (such as title, date written, etc.) and one field for the complete content of a single work. An alternative view, defined over the base relation or against the first view, might have one tuple for each scene in each play, with attributes for play title, act number, scene number, and content. By supporting structured text matching and providing conversion operators to create relational views of selected portions of text, the full relational operator set is available to applications, providing flexibility, expressiveness, and efficiency.

5 Conclusions

Although text appears to be a simple type, it can have a rich structure. Simple decisions, such as whether "words" are fundamental units, influence how widely a text model can be applied.

The fact that text is created primarily for communications among humans results in variability among text values that requires exceptional flexibility to accommodate it. Even for highly stylized texts, such as an encyclopedia or dictionary, variations in expression among seemingly similar items is commonplace; rigid conformation to detailed schemas is rare.

We have designed and implemented a federated database system that manages text data alongside more conventional relational data[2]. We have found that flexible fulltext search can then be used in conjunction with tree operators that address the structure of text, and that those two aspects can be accommodated within an extended SQL. Our language extensions have been accepted for progression by ISO's SQL/MM committee[7], and a prototype system demonstrating these language features in a federated database environment has been available over the Web for approximately one year[24].

Acknowledgements

The ideas in this paper have evolved from discussions with many colleagues associated with the University of Waterloo Centre for the New OED and Text Research, including G. E. Blake, T. Bray, L. J. Brown, M. P. Consens, I. J. Davis, G. H. Gonnet, P. Kilpeläinen, E. Kuikka, P.-Å. Larson, S. Li, K.-Y. Ng, C. R. Palmer, D. R. Raymond, A. Salminen, and T. Snider. Financial support for the research has been provided by the Natural Sciences and Engineering Research Council of Canada, Industry Canada (through the Canadian Strategic Software Consortium), Open Text Corporation, and the University of Waterloo.

References

[1] G. E. Blake, T. Bray, and F. W. Tompa, "Shortening the *OED*: experience with a grammar-defined database," *Trans. Information Systems 10*, 3 (July 1992), 213–232.

[2] G. E. Blake, M. P. Consens, I. J. Davis, P. Kilpeläinen, E. Kuikka, P.-Å. Larson, T. Snider, and F. W. Tompa, "Text / Relational Database Management Systems: Overview and Proposed SQL Extensions," Dept. of Computer

Science Technical Report CS-95-25, University of Waterloo (June 1995) 31 pp.

[3] L. J. Brown, M. P. Consens, I. J. Davis, C. R. Palmer, and F. W. Tompa, "A Structured Text ADT for Object-Relational Databases," Dept. of Computer Science Technical Report CS-97-26, University of Waterloo (July 1997) 24 pp.

[4] T. Bray and C. M. Sperberg-McQueen, editors, "Extensible Markup Language (XML): Part I. Syntax ," W3C Working Draft WD-xml-lang-970331 (March 1997).

[5] C. L. A. Clarke, G. V. Cormack, and F. J. Burkowski, "Fast Inverted Indexes with Online Update," Dept. of Computer Science Technical Report CS-94-40, University of Waterloo (November 1994) 11 pp. See also http://multitext.uwaterloo.ca.

[6] J. H. Coombs, A. H. Renear, S. J. de Rose, "Markup systems and the future of scholarly text processing", *Comm. ACM 30*, 11 (November 1987), 933–947.

[7] I. J. Davis, "Adding structured text to SQL/MM Part 2: Full Text, A change proposal," ISO/IEC JTC1/SC21/WG3 CAC N334R3, April 26, 1996.

[8] W. B. Frakes and R. Baeza-Yates, editors, *Information Retrieval: Data Structures & Algorithms.* Prentice Hall, 1992, 504 pp.

[9] T. Gaasterland, *et al.*, editors, *Proc. 5th Int. Conf. on Intelligent Systems for Molecular Biology, ISMB-97* Halkidiki, Greece (June 21-25, 1997), AAAI Press, 375 pp.

[10] C.F. Goldfarb, *The SGML Handbook*, Oxford University Press, Oxford, U.K., 1990.

[11] G. H. Gonnet and F. W, Tompa, "Mind your grammar: a new approach to modelling text," *Very Large Data Bases* (VLDB), Vol. 13 (September 1987), 339–346.

[12] G. H. Gonnet, R. A. Baeza-Yates, and T. Snider. "New indices for text: PAT trees and PAT arrays," in Frakes and Baeza-Yates [8], 1992, 66–82.

[13] G. H. Gonnet, "A Tutorial Introduction to Computational Biochemistry Using Darwin," E.T.H. Zürich, Switzerland, 1994.

[14] International Organization for Standardization, *ISO/IEC JTC1/SC21 Information Retrieval, Transfer and Management for OSI WG3 Database*: ISO/IEC SC21/WG3 N1679 SQL/MM SOU-004 ISO Working Draft SQL Multimedia and Application Packages (SQL/MM) — Part 2: Full-Text, March 1994.

[15] P. Kilpeläinen and H. Mannila, "Retrieval from hierarchical texts by partial patterns," *16th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval* (1993) 214-222.

[16] L. Lamport, *A Document Preparation System: LaTeX, 2nd edition*, AMS and Addison-Wesley, 1994.

[17] S. Li, "Using Inverted Lists to Match Structured Text," Dept. of Computer Science Technical Report CS-96-34, University of Waterloo (August 1996) 93 pp.

[18] I.A. Macleod, "Data modelling requirements for document," *Proc. IFIP TC8/WG8.1 Conference: Information Systems Concepts: Improving the Understanding*, North-Holland, 1992, 259–272.

[19] K.-Y. Ng, "The Use of a Combined Text/Relational Database System to Support Document Management," Dept. of Computer Science Technical Report CS-96-07, University of Waterloo (1996).

[20] D. R. Raymond, "Flexible text display with Lector," *IEEE Computer 25*, 8 (August 1992), 49–60.

[21] A. Salminen and F. W. Tompa, "PAT expressions: an algebra for text search," *Acta Linguistica Hungarica 41*, 1–4 (1992–93) 1994, 277–306.

[22] A. Salminen and F. W. Tompa, "Grammars++ for Modelling Information in Text," Dept. of Computer Science Technical Report CS-96-40, University of Waterloo (November 1996) 46 pp.

[23] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill Computer Science Series. McGraw-Hill, NY, 1983.

[24] The TRDBMS Project, "Integrating Structured Text and SQL," The Centre for the New OED and Text Research, Dept. of Computer Science, University of Waterloo, `http://solo.uwaterloo.ca/trdbms/`.

[25] O. Yasishi and I. Masajirou, "A new character-based indexing method using frequency data for Japanese documents," *ACM SIGIR 95*, 261–275.