EMU: AN E-MAIL PREPROCESSOR FOR TEXT-TO-SPEECH

R. Sproat, J. Hu
Bell Labs, Lucent Technologies
Murray Hill, NJ

H. Chen
University of California
Berkeley, CA

Abstract - E-mail reading is one of the most important commercial applications of text-to-speech synthesis (TTS). Yet e-mail is one of the most difficult types of text to deal with, since it is both highly structured — frequently containing elements such as tables, signatures, "artwork" and quotations from previous messages; and at the same time often lacks any reliable unambiguous indicators for such structure. This paper describes *Emu*, an *e*-mail *m*ark-*u*p and rendering program that preprocesses e-mail for TTS. We discuss algorithms for detecting regions of interest in the input text; for "normalizing" the input; and for actually rendering the input through the Bell Labs TTS system.

INTRODUCTION

Audio rendering of e-mail is one of the most important commercial applications of text-to-speech synthesis (TTS), and is a central component of multimedia messaging. Yet e-mail is one of the most difficult types of text to deal with, since it is both highly structured — frequently containing elements such as tables, signatures, "artwork" and quotations from previous messages; and at the same time often lacks any reliable unambiguous indicators for such structure. This paper describes *Emu*, an *e*-mail *m*ark-*u*p and rendering program that preprocesses e-mail for TTS.

It is useful to draw an analogy between the process of converting an electronic document into a printed rendering of that document, and the process of converting a document into speech — what T. V. Raman [1] terms an *audio rendering*. Consider first a printed document: the printed version of this paper was produced by first preparing a document with various structural properties *marked up* using LaTeX commands; second a *device independent* rendering was computed from which a variety of different output formats can be derived; and third, the device-independent representation was converted into Postscript. This final form can then be displayed or printed on any postscript device.

The model assumed by Emu is similar. An e-mail message is first parsed into different regions (headers, quoted material, signature blocks ...), and these regions are marked with tags that indicate the regions' properties. Second a normalization of the text is computed. The normalization performed in this second phase largely involves the expansion of unusual "words", including "acronyms" (*WinNT*), as well as e-mail addresses, URL's and other non-standard material. The output of the normalization phase is "device independent" in the sense that the normalizations performed produce text that is appropriate as input to any (English) TTS system. Finally, in the third phase, the marked-up and normalized text is *rendered* by converting it into text interspersed with control sequences for the Bell Labs American English TTS system [2].

ALGORITHMS FOR TEXT-REGION DETECTION

The first phase of analysis performed by Emu is the detection and markup of significant portions of the input text. Emu assumes that the input text is blocked into regions, where a region is defined to be a block of contiguous lines separated from other regions by either: (i) one or more blank lines; or (ii) a clearly defined separator, such as a forwarded message delimiter. Each region is assumed to be of the same type. So, for example, a region may be tagged as plain text (PTEXT) or as a table (TABLE), but not both. This uniformity assumption, while not absolutely correct, is nonetheless correct often enough to be useful. Note that if one were not to make this assumption, analyzing the input text would become significantly more computationally expensive, since one would have to consider every block of contiguous lines to be a potential region.

In some cases the detection of a region is straightforward. For example, it is relatively easy to detect e-mail headers since they are readily identifiable by the existence of certain line-initial tags, such as *From:*, *To:*, *Subject:* and so forth. Detection of other region types requires more sophisticated analysis. Emu distinguishes among eight basic text-region types: plain text (PTEXT), "artwork" (GRAPHIC), quoted regions (QUOTED), itemized lists (ITEM), signatures (SIG), headlines (HEADL), addresses (ADDRESS) and tables (TABLE).[1]

The initial detection of these regions starts with the following intuition: for many types of regions, one does not need to be able to see the text clearly in order to decide that the region is of a particular type. So, imagine that one were looking at a page from a distance, so that it is impossible to read what is on the page. Nonetheless one could clearly detect tables, many signatures, addresses, headlines and graphics by the mere appearance of the text. Each of these region types has a fairly reliable distribution of real text material and whitespace. So headlines tend to be short regions of text where the material is centered on the page. Tables and many signatures tend to have the text arranged in columns with the columns separated by a fair amount of whitespace. Addresses tend to consist of a series of short lines. Finally

[1]This particular 8-way classification was suggested to us by David Yarowsky, who also kindly provided us with some initial hand-tagged training data.

graphics tend to have a lot of whitespace and only sparsely distributed non-whitespace material. Plain text, quoted text and itemized list elements are of course more difficult to detect by such coarse means, and more special-purpose techniques are needed in these cases. For example, quoted regions are frequently marked with one of a handful of distinguished characters (often '>') at the beginning of each line.

The basic intuition outlined above is implemented as follows. First each character in the input is deterministically mapped to one of a set of predefined character classes. The character classes currently used include: SPACE, DIGIT, ALPHABETIC, '>' and ':' (common indicators of quoted messages), and PUNCTUATION. Second, the individual character-class-encoded lines of each block are matched against a set of eight *character class pentagram* models, one for each text region type. These models are trained on 5,590 lines of hand-tagged netnews text. Applied to each line, the models give a measure, for each text region type, of how strongly the given line matches that particular type. Formally, the models are implemented as *weighted finite-state acceptors* — WFSA's [3], and each of the models is intersected with the input line, itself represented as an unweighted acceptor. This gives us a weight for each of the class assignments for each line. Of course, what we want is a unique classification for each line, and a uniform classification for the block. The third phase of the analysis accomplishes this. The block is represented as a weighted finite-state automaton with $n+1$ states, where n is the number of lines in the block. There are eight arcs between each pair of adjacent states i and $i+1$, each labeled with one of the eight character classes, and weighted according to the pentagram score for the ith line for that class. An example of such an automaton is given in Figure 1. This automaton — henceforth B for *block automaton* — is then combined with a set of finite-state acceptors — G — implementing grammatical restrictions, such as the restriction that all elements of a block must be of the same type. Also implemented are some length restrictions: for example, ADDRESS fields are rarely shorter than two lines; and SIG fields are rarely longer than ten lines. The analysis A of the block is obtained by intersecting B with G, and then computing the lowest cost path. Formally: $A = \mathrm{BestPath}(B \cap G)$.

The performance of the algorithm just described was measured on a test corpus of 2543 lines of netnews text. Overall performance, counting by correctly classified lines (ignoring blanks), had an error rate of 7%. The largest class of errors involves SIG fields that are misclassified as PTEXT. As we shall see momentarily, potential SIG fields — including material near the end of the message initially classified as PTEXT — undergo a more rigorous analysis that corrects some of the errors.

Once the classification phase is complete, the tagged blocks are parsed into a document tree, with the node dominating each block tagged with the class computed for that block: the model of document structure assumed by Emu follows closely that of SGML [4].

Two types of blocks then receive further treatment: QUOTED regions

```
[0]    Charles Davies
[1]    Dialogue Modeling Research Department
[2]    Multimedia Communications Research Laboratory
[3]    Bell Laboratories, Lucent Technologies    | tel (908) 582-1234
[4]    600 Mountain Avenue, Room 2d-500          | fax (908) 582-4321
[5]    Murray Hill, NJ 07974, USA                | ced@bell-labs.com
[6]    http://www.bell-labs.com/noname/mcs/
```

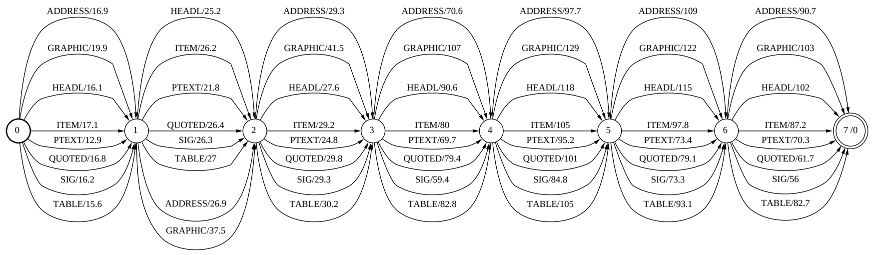

Figure 1: Block automaton and its associated SIG region. Arcs exiting state i correspond to input line i.

```
Charles Davies                  Email:  ced@research.bell-labs.com
Lucent Technologies Bell Labs   WWW: http://www.bell-labs.com/noname/mcs/
600 Mountain Avenue, 2D-500     Voice:  908 582-1234
Murray Hill, NJ 07974           Fax:  908 582-4321
```

Figure 2: A block with two connected components.

are recursively subjected to the classification and parsing algorithm just described; and potential signature blocks undergo the finer-grained analysis described next. The signature block analysis starts with a two-dimensional geometrical analysis that attempts to find connected components. For example, the signature block in Figure 2 consists of two connected components, corresponding to the two columns. The geometrical analysis yields a set of *reading blocks* corresponding to the discovered connected components: in general, it is assumed that when reading the text in the signature, elements that are part of the same reading block belong more naturally together than elements in different reading blocks. But within a reading block there may be several *functional blocks*. For example, in Figure 2, the righthand reading block contains an e-mail address, some WWW contact information, a phone number and a fax number. Each of these constitutes a separate functional block. Functional blocks are detected using language analysis of the text material within each reading block. If the signature-block analysis algorithms detect certain required components — e-mail addresses, web addresses, phone or fax numbers, names or postal addresses — then the algorithms proceed with the complete analysis and mark the signature region as “verified”. Otherwise they mark the region as unverified and return it to the parsing phase of Emu. Emu will then revert the region to its original analysis, or if it was originally assigned to SIG, it will be reverted to PTEXT. Further details of the signature block analysis algorithms can be found in [5].

DEVICE-INDEPENDENT RENDERING

The function of Emu’s device-independent rendering phase is to “normalize” various elements in the marked up document to improve the final rendering into speech. Among the functions that this phase performs are:

- Detect and mark “separator” lines, such as the “--” commonly found at the beginning of SIG fields.
- Expand *[Rr]e:* in the *Subject:* line into *regarding*.
- Detect various conventions for marking emphasis: conventions include capitalization, and delimiting the word with asterisks (**require**). Emphatic words are marked with the tag <emph>.
- Expand emoticons (smileys) into words.
- Decide whether capitalized words should be read as words (*LOS ANGELES*, *UNESCO*) or as sequences of letters (*CIA*, *WABC*).
- Expand electronic addresses, pathnames, and URL’s appropriately.
- Expand non-conventional words such as *WinNT* into appropriate renderings.

For reasons of space, we cannot describe how all of these types of analysis are performed. Instead we describe the handling of one problem, namely the treatment of capitalized words. We use a simple, but surprisingly effective algorithm to determine whether or not to read a capitalized word as a word, or as a sequence of letters:

- If the capitalized sequence is longer than 5 letters, read it as a word.
- If the capitalized sequence is 5 letters or shorter, check the word against a dictionary of known words of 5 letters or shorter. If it is in this dictionary read it as a word. Otherwise read it as a sequence of letters.

The current dictionary, derived from words in the 1994 Associated Press (AP) newswire, 5 letters or shorter, contains about 11,000 entries. For words longer than five letters, for example, the algorithm is correct over 99% of the time, as measured on the January 1994 AP.

Naturally, such disambiguation cannot always reliably be done in a context-independent fashion: one cannot tell whether *ADA* should be read as a word (i.e., the name *Ada*) or as a sequence of letters (i.e., as an abbreviation for the *American Dental Association*), without considering the context in which the word occurs. Given time, disambiguation methods discussed in [6] could be brought to bear on this problem. But as a first-pass model, the present one is quite effective.

AUDIO RENDERING

Audio rendering is the phase in which Emu performs its final formatting of the input text, and passes it off to the TTS system for conversion into speech. As noted above, the marked-up and normalized text is *rendered* by converting it into text interspersed with controls for the Bell Labs TTS system. The most important function of the controls in this application are to change the voice used by the synthesizer as a means of indicating the document structure. Thus, when the system renders an embedded quotation, for example, it changes to a different voice, staying in that voice until the end of the quotation, at which point it reverts to the previously used voice. (The user is also explicitly informed that the system has entered a new level of structure. For example, by default a top-level quoted message will result in the system informing the user: "this is a level 1 quoted message".) For some types of text element Emu does little more than inform the user that there is a region of that type present. For a GRAPHIC, for example, Emu informs the listener that "there is a region of non-text of type ascii-graphic here."

A demo example showing the behavior of Emu can be found at the following URL: `http://www.bell-labs.com/project/tts/emu.html`.

References

[1] T.V. Raman. *Audio System for Technical Readings.* PhD thesis, Cornell University, 1994.

[2] Richard Sproat, editor. *Multilingual Text-to-Speech Synthesis: The Bell Labs Approach.* Kluwer, Boston, MA, 1997.

[3] Mehryar Mohri, Fernando Pereira, and Michael Riley. A rational design for a weighted finite-state transducer library. In *Second International Workshop on Implementing Automata*, pages 43–53, Ontario, Canada, september 1997.

[4] Charles Goldfarb. *The SGML Handbook.* Clarendon Press, Oxford, 1990.

[5] Hao Chen, Jianying Hu, and Richard Sproat. E-mail signature block analysis. In *ICPR'98*, Brisbane, Australia, August 1998.

[6] David Yarowsky. *Three Machine Learning Algorithms for Lexical Ambiguity Resolution.* PhD thesis, University of Pennsylvania, 1996.