Documentation for Stage One 10-X Parse

1. Author: Professor Bill McDonald
Mendoza College of Business
University of Notre Dame
Notre Dame, IN 46556
mcdonald.1@nd.edu

2. Stage One Parse

All 10-X SEC complete text document filings are downloaded for each year/quarter. We use "10-X" to represent any SEC filing that is a 10-K variant, e.g., 10-Q, 10-K/A, 10-K405, etc.[1] The text version of the filings provided on the SEC server is an aggregation of all information provided in the browser-friendly files also listed on Edgar for a specific filing. For example, IBM's 10-K filing on 20120228 lists the core 10-K document in html format, ten exhibits, four jpg (graphics) files, and six XBRL files.[2] All of these files are also contained in a single text file with the embedded HTML, XBRL, exhibits, and the ASCII-encoded graphic.[3] In the IBM example, of the 48,253,491 characters contained in the file, only about 7.6% account for the 10-K text including the exhibits and tables. The HTML coding accounts for about 55% of the file. The XBRL tables have a very high ratio of tags to data and account for about 33% of the text file. The remaining 27% of the file is attributable to the ASCII-encoded graphics. In many cases, ASCII-encoded pdfs, graphics, xls, or other binary files that have been encoded can account for more than 90% of the document.

Because most textual analysis studies focus on the textual content of the document, we have created files where all of the 10-X documents have been parsed to exclude markup tags, ASCII-encoded graphics, and tables. We exclude tables, because they are usually not the focus of textual analysis. Certainly, one can imagine research where any of these excluded items might require analysis of the original filings, however the compressed files provide for a standardized and efficient way of facilitating those studies focused on text.

---

[1] To ensure that all relevant filings are included, the download process is actually overly generous and includes any filing with "10" included in the filename. This assures that we capture all of the variations in annual and quarterly filings. For example the downloaded files include 10-12B filings (Initial general form for registration of a class of securities pursuant to section 12(b)), and mislabeled filings, such as NBG Radio (CIK=1059366, FilingDate=20020228) where they file their "10KSB" as "10-KSB".
[2] XBRL (eXtensible Business Reporting Language) is a markup language. A variant of XML and related to HTML, it provides semantic context for data reported within a 10-K. For example, one line in Google's 20111231 10-K filing contains "<us-gaap:StockholdersEquity contextRef="eol_PE633170--1110-K0018_STD_0_20081231_0" unitRef="iso4217_USD" decimals="-6">28239000000</us-gaap:StockholdersEquity>". The "eol …" segment defines the XBRL implementation, the data are in US dollars and the "-6" indicates the number is rounded to millions. See http://xbrl.sec.gov. A few firms began including XBRL in their filings in 2005 with the number expanding substantially in 2010.
[3] ASCII-encoding converts binary data files to plain ASCII-printable characters, thus ensuring cross platform conformity. The conversion from binary to plain text increases the size of the original file by orders of magnitude.

3. Markup Tags

All of the original markup language tags (HTML, XBRL, XML) are deleted from the original document. We insert our own markup tags within a header at the beginning of the compressed document and tags to delineate all exhibits in the document. The structure of the tagging system is as follows:

3.1. The following information appears at the beginning of each file:

<Header>
<FileStatsLabels>GrossFIleSize,NetFileSize,ASCIIEncodedChars,HTMLChars,XBRLChars,TableChars</FileStatsLabels>
<FileStats> … *comma delimited data as labeled above* … </FileStats>
<SEC-Header> … *All text contained in the original SEC-Header* </SEC-Header>
</Header>

Note that the FileStats data contain character counts for the size of the raw file, the post-parsing size, and character counts for the items deleted from the document.

3.2 All exhibits preceded by the original tags of "<TYPE>EX-##" are encapsulated in the parsed files as:

<EX-##>
… original text
</Exhibit>

4. Parsing Details

Each raw text file downloaded from EDGAR is parsed using the following sequence. Relevant Regular Expression code is provided in parentheses.

5.1. Remove ASCII-Encoded segments – All document segment <TYPE> tags of GRAPHIC, ZIP, EXCEL and PDF are deleted from the file. ASCII-encoding is a means of converting binary-type files into standard ASCII characters to facilitate transfer across various hardware platforms. A relative small graphic can create a substantial ASCII segment. Filings containing multiple graphics can be orders of magnitude larger than those containing only textual information.
5.2. Remove <DIV>,<TR>,<TD> and <FONT> tags – Although we require some HTML information for subsequent parsing, the files are so large (and processed as a single string) that we initially simply strip out some of the formatting HTML.
5.3. Remove all XBRL – all characters between <XBRL …> … </XBRL> are deleted.
5.4. Remove SEC Header/Footer – All characters from the beginning of the original file thru </SEC-HEADER> (or </IMS-HEADER> in some older documents) are deleted from the

file. Note however that the header information is retained and included in the tagged items discussed in section 4.1. In addition the footer "-----END PRIVACY-ENHANCED MESSAGE-----" appearing at the end of each document is deleted.

5.5. Remove tables – all characters appearing between <TABLE> and </TABLE> tags are removed.

  5.5.1. Note that some filers use table tags to demark paragraphs of text, so each potential table string is first stripped of all HTML and then the number of numeric versus alphabetic characters is compared. For this parsing, only table encapsulated strings where *numeric chars/(alphabetic+numeric chars)* > 15%.

  5.5.2. In some instances, Item 7 and/or Item 8 of the filings begins with a table of data where the Item 7 or 8 demarcation appears as a line within the table string. Thus, any table string containing "Item 7" or "Item 8" (case insensitive) is *not* deleted.

5.6. Remove all carriage returns – EDGAR files use a linefeed (/n) as the "newline" character. Because of the way we process very large files the actual string we parse can contain the Windows end-of-line specification (i.e., carriage return followed by a linefeed -- \r\n). In order to simplify subsequent parsing all carriage returns are deleted from the document, i.e., all documents will have only a linefeed as the "newline" character.

5.7. Tag Exhibits – At this point in the parsing process all exhibits are tagged as discussed in section 4.2.

5.8. Remove Markup Tags – remove all remaining markup tags (i.e., <…>).

5.9. Re-encode reserved HTML characters (character entity references)—In order to encode a broad set of universal characters within the limitations of ASCII coding many characters are encoded. For example, the "&" symbol can be encoded as "&amp;" or "&#38;". For items 5.9.1.-5.9.6 listed below we replace the encode items with a character(s). The remaining encoded items are deleted.

  5.9.1. "&LT;" or "&#60" -> " LT " - note we use LT instead of "<" to avoid any confusion with markup tags.

  5.9.2. "&GT;" or "&#62" -> " GT "

  5.9.3. "&NBSP;" or "&#160;" -> " "

  5.9.4. "&QUOT;" or "&#34" -> """

  5.9.5. "&APOS;" or "&#39" -> "'"

  5.9.6. "&AMP;" or &#38" -> "&"

  5.9.7. All Regular Expression \t and \v items are deleted.

  5.9.8. All remaining ISO 8859-1 symbols and characters are deleted.

5.10. Finally some remaining idiosyncratic anomalies are parsed out:

  5.10.1. Linefeeds (\n) following hyphens are removed.

  5.10.2. Hyphens preceded and followed by a blank space are removed.

  5.10.3. The token "and/or" (case insensitive) is replaced by "and or".

  5.10.4. Sequences of two or more hyphens, periods or equal signs possibly followed by spaces (e.g., REGEX = "(-|\.|=)\s*") are removed.

  5.10.5. All underscore characters ("_") are removed.

  5.10.6. All sequences of three or more blanks are replaced by a single blank.

5.10.7. All sequences of three or more linefeeds possibly separated by spaces (REGEX = “(\n\s*){3,}”) are replaced by two linefeeds.

5.10.8. All linefeeds not preceded by a linefeed and not followed by a blank or linefeed are replaced by a blank.