TikaEvalDbDesign

Design of the Tika Eval Database

Background

The tika-eval module was initially designed to work with the output from Tika's RecursiveParserWrapper (`-J -t` in tika-app-ese or `/rmeta/text` from tika-server). The basic idea is that for each input file, there is a list of Metadata objects. The first Metadata object in the list contains the information from the outer/container input file, and if there are attachments or embedded files, those are added to the list. The content that was extracted is stored in the "X-TIKA:content" key in each Metadata object.

In the following discussion, we use the term "container" to refer to the initial input file, and we use "file" to refer either to the container file or to an embedded object. An "extract" is the file that contains the extracted metadata/content.

Because of this one->many mapping, the table structure became fairly complex fairly quickly. The initial flat file didn't work so well.

Tables

Profile

- PROFILES – this is the main table. This contains all files, embedded and not, whether or not there was an exception. The primary key is ID, and this table has foreign keys to the CONTAINERS table (CONTAINER_ID), and to the MIMES table (MIME_TYPE_ID). This includes a boolean value for whether or not this file was an embedded file or a container file (IS_EMBEDDED). Note that the ID==CONTAINER_ID for the container file; this makes some useful joins much easier.
- CONTAINERS – this contains information about the container file. The primary key is CONTAINER_ID and that maps to the PROFILES' table's CONTAINER_ID foreign key.
- CONTENTS – this contains statistics about the content that was extracted. The primary key is ID and this maps directly to the PROFILES' table's ID.
- MIMES – this contains mime strings and the most common file extension for each mime type. The primary key is MIME_TYPE_ID, and this maps to the MIME_TYPE_ID column in the PROFILES table.
- PARSE_EXCEPTIONS – this contains information about parse exceptions. The primary key is ID and that maps to the ID column in the PROFILES table. This includes the original stack trace, a slightly truncated stack trace used for sorting and grouping by, and a foreign key to the class/type of exception.
- REF_PARSE_EXCEPTION_TYPES – this is a reference table that contains the different classes/types of parse exceptions: runtime, encryption (password protected files), access_permission (e.g. a PDF that won't allow extraction of text) and unsupported version. We have custom code that maps stacktraces to these exception types. Let us know if you find any problems!
- ERRORS – this contains records for errors reading the extract file or errors that were caused during the parse that caused tika-app in batch mode to restart. By mapping to the REF_EXTRACT_ERROR_TYPES and the REF_PARSE_ERROR_TYPES, you can identify the cause of the problem. There's a todo to integrate the code that reads in the parser error log and update this table...that functionality is on its way.
- EMB_FILE_NAMES – this records the full path of an embedded file **within** the file. If there's a zip with a zip in it, you might see: /zip1.zip/zip2.zip /my_docx.docx. Again, the ID in this table maps to the ID in the PROFILES table.

Comparison

Nearly all of the above exist with '_A' or '_B' added to the end of the table names. The ID in PROFILES_A matches the ID in PROFILES_B. If a given container file contained a different number of attachments or if the code couldn't figure out which attachments map to which attachments, there can be incorrect double entries between the two.

In addition to the profiling tables, there is also:

- CONTENT_COMPARISONS – this records the output of comparing the content. Again, the ID maps to the ID in PROFILES_A and PROFILES_B.

Columns

CONTENTS

- CONTENT_LENGTH – literal string length
- NUM_TOKENS – number of tokens (as analyzed by the Lucene analysis chain)
- NUM_UNIQUE_TOKENS – number of unique tokens (also known as "types")
- COMMON_TOKENS_LANG – which language was selected for counting "common tokens"
- NUM_COMMON_TOKENS – number of "common tokens" (not number of unique tokens) that were found
- TOP_N_TOKENS – top 10 most frequent tokens
- NUM_ALPHABETIC_TOKENS – number of tokens that contained at least one alphabetic/ideographic character
- LANG_ID_1 – language id as determined by Optimaize
- LANG_ID_1_PROB_1 – language id probability.
- LANG_ID_2, etc. if Optimaize thought there could be more than one language classification.
- UNICODE_CHAR_BLOCKS – number of characters per Unicode code block
- TOKEN_ENTROPY_RATE – token entropy, a measure of diversity of terms. A word list would have a very high entropy, a novel by someone with a 10 word vocabulary would have a very low entropy rate.

- TOKEN_LENGTH_SUM, *_MEAN, *_STD_DEV – sum, mean, stdev of the lengths of the tokens

CONTENT_COMPARISONS

- TOP_10_UNIQUE_TOKEN_DIFFS_A – what are the top 10 most frequent terms that appear in A and never appear in B.
- TOP_10_UNIQUE_TOKEN_DIFFS_B – *vice versa*
- TOP_10_MORE_IN_A – the top 10 terms that appear more frequently in A than B. The number is the difference between A and B per term (e.g. you'd see `"cat": 10` if "cat" appeared 30 times in A and 20 times in B)
- TOP_10_MORE_IN_B – *vice versa*
- DICE_COEFFICIENT – focuses on the overlap based on unique tokens (not tokens): 2*overlap/(totalUniqueTokensA+totalUniquTokensB)
- OVERLAP – focuses on the overlap of tokens: (2*overlap)/(totalTokensA+totalTokensB)