TikaEval

Overview of the 'tika-eval-app' Module

This page offers a first draft of the documentation for the tika-eval-app module, which was initially added to Tika 1.15.

The module is intended to offer insight from the output of a single extraction tool or to enable some comparisons between tools. This module is designed to be used to help with Tika, but it could be used to evaluate other tools as well.

As part of Tika's periodic regression testing, we run this module against ~3 million files (for committers/PMC interested in running the regression testing on our regression vm, see TikaEvalOnVM). However, it will not scale to 100s of millions of files as it is currently designed. Patches are welcomed!

Background

There are many tools for extracting text from various file formats, and even within a single tool there are usually countless parameters that can be tweaked. The goal of 'tika-eval' is to allow developers to quickly compare the output of:

1. Two different tools
2. Two versions of the same tool ("Should we upgrade? Or are there problems with the newer version?")
3. Two runs with the same tool but with different settings ("Does increasing the DPI for OCR improve extraction? Let's try two runs, one with 200 DPI and one with 300")
4. Different tools against a gold standard

In addition to this "comparison mode", there is also plenty of information one can get from looking at a profile of a single run.

Some basic metrics for both the "comparison" and "profiling" mode might include:

- Exceptions – how many and of what category? Are these regular catchable exceptions, evil OOMs or permahangs?
- Metadata – how many metadata values did we extract?
- Embedded files/attachments – how many embedded files were found
- Mime detection – how many of what types of files do we have? Where do we see discrepancies between tools?
- Content – is the content extracted by tool A better than that extracted by tool B? On which files is there a big difference in extracted content?

The tika-eval module was initially developed for text only. For those interested in evaluating structure/style components (e.g. <title/> or <b/> elements), see TikaEvalAndStructuralComponents.

Quick Start Usage

NOTE: tika-eval will not overwrite the contents of the database you specify in Profile or Compare mode. Add `-drop` to the commandline to drop tables if you are reusing the database.

The following assumes that you are using the default in-memory H2 database. To connect tika-eval to your own db via jdbc see TikaEvalJdbc.

Single Output from One Tool (Profile)

NOTE: assume the original input files are in a directory named `input_docs` and that the text extracts are written to the `extracts` directory, with each extract file having the same sub-directory path and same file name with '.json' or '.txt' appended to it.

1. Create a directory of extract files that mirrors your input directory. These files may be UTF-8 text files with '.txt' appended to the original file's name or they may be the RecursiveParserWrapper's '.json' representation: `java -jar tika-app-X.Y.jar -J -t -i input_docs -o extracts`
2. Profile the directory of extracts and create a local H2 database:
   ```
   java -jar tika-eval-X.Y.jar Profile -extracts extracts -db profiledb
   ```
3. Write reports from the database:
   ```
   java -jar tika-eval-X.Y.jar Report -db profiledb
   ```

You'll have a directory of .xlsx reports under the "reports" directory. **Note:** if you don't need the full tika-eval-app, you can get many of these statistics at parse time via the TikaEvalMetadataFilter (see: ModifyingContentWithHandlersAndMetadataFilters).

Comparing Output from Two Tools/Settings (Compare)

NOTE: assume the original input files are in a directory named `input_docs` and that the text extracts from tool A are written to the `extractsA` directory and the extracts from tool B are written to `extractsB`.

1. Create two directories of extract files that mirror your input directory. These files may be UTF-8 text files with '.txt' appended to the original file's name or they may be the RecursiveParserWrapper's '.json' representation.
2. Compare the extract directory A with extract directory B and write results to a local H2 database:
   ```
   java -jar tika-eval-X.Y.jar Compare -extractsA extractsA -extractsB extractsB -db comparisondb
   ```
3. Write reports from the database:
   ```
   java -jar tika-eval-X.Y.jar Report -db comparisondb
   ```

You'll have a directory of .xlsx reports under the "reports" directory.

Investigating the Database

1. Launch the H2 localhost server:
   `java -jar tika-eval-X.Y.jar StartDB` – this calls `java -cp ... org.h2.tools.Console -web`
2. Navigate a browser to `http://localhost:8082` and enter the jdbc connector code followed by the **full path** to your db file:
   `jdbc:h2:/C:/users/someone/mystuff/tika-eval/comparisondb`

If your reaction is: "You call this a database?!", please open tickets and contribute to improving the structure.

See TikaEvalDbDesign for more information on the underlying structure of the database.

More detailed usage

Evaluating Success via Common Words

In the absence of ground truth, it is often helpful to count the number of common words that were extracted (see TikaEvalMetrics for a discussion of this).

"Common words" are specified per language in the "resources/commonwords" directory.
Each file is named for the language code, e.g. 'en', and each file is a UTF-8 text file with one word per line.

The token processor runs language id against content and then selects the appropriate set of common words for its counts. If there is no common words file for a language, then it backs off to the default list, which is currently hardcoded to 'en'.

Make sure that your common words have gone through the same analysis chain as specified by the Common Words analyzer in 'lucene-analyzers.json'!

Reading Extracts

alterExtract

Let's say you want to compare the output of Tika to another tool that extracts text. You happen to have a directory of .json files for Tika and a directory of UTF-8 .txt files from the other tool.

1. If the other tool extracts embedded content, you'd want to concatenate all the content within Tika's .json file for a fair comparison:
   `java -jar tika-eval-X.Y.jar Compare -extractsA tika_1_14 -extractsB tika_1_15 -db comparisondb -alterExtract concatenate_content`
2. If the other tool does not extract embedded content, you'd only want to look at the first metadata object (representing the container file) in the .json file:
   `java -jar tika-eval-X.Y.jar Compare -extractsA tika_1_14 -extractsB tika_1_15 -db comparisondb -alterExtract first_only`

Min/Max Extract Size

You may find that some extracts are too big to fit in memory, in which case use `-maxExtractSize <maxBytes>`, or you may want to focus only on extracts that are greater than a minimum length: `-minExtractSize <minBytes>`.

Reports

The module tika-eval comes with a list of reports. However, you might want to generate your own. Each report is specified by SQL and a few other configurations in an xml file. See `comparison-reports.xml` and `profile-reports.xml` to get a sense of the syntax.

To specify your own reports on the commandline, use `-rf` (report file):
```
java -jar tika-eval-X.Y.jar Report -db comparisondb -rf myreports.xml
```

If you'd like to write the reports to a root directory other than 'reports', specify that with `-rd` (report directory):
```
java -jar tika-eval-X.Y.jar Report -db comparisondb -rd myreportdir
```

Again, see TikaEvalDbDesign for more information on the underlying structure of the database.