MTR140443R2
MITRE TECHNICAL REPORT

MITRE

Methods for Evaluating Text Extraction Toolkits: An Exploratory Investigation

Timothy B. Allison
Paul M. Herceg
January 22, 2015

Contract No.: W15P7T-13-C-A802
Project No.: 0714G01Z-HB

MITRE Department
And Project Approval:

Jeffrey M. Siems, Project Leader

Methods for Evaluating Text Extraction Toolkits: An Exploratory Investigation

Timothy B. Allison, Paul M. Herceg

The MITRE Corporation, 7515 Colshire Drive, McLean, VA 22102, USA

Abstract

Text extraction tools are vital for obtaining the textual content of computer files and for using the electronic text in a wide variety of applications, including search and natural language processing. However, when extraction tools fail, they convert once reliable electronic text into garbled text, or no text at all. The techniques and tools for validating the accuracy of these text extraction tools are conspicuously absent from academia and industry. This paper contributes to closing this gap. We discuss an exploratory investigation into a method and a set of tools for evaluating a text extraction toolkit. Although this effort focuses on the popular open source Apache Tika toolkit and the *govdocs1* corpus, the method generally applies to other text extraction toolkits and corpora.

Keywords: validation; evaluation; text extraction; text extraction toolkits; computer files; electronic text; reliable electronic text; Tika; Apache; garbled text; mojibake; content extraction

1. Introduction

Numerous applications need access to the normalized textual content of computer files, including Web and corporate search engines, email filters, parental control software, applications for the blind, smart phone applications allowing hands-free operation, summarization tools, information extraction engines, transliteration tools, and machine translation engines [1]. These applications need *reliable electronic text*—a topic that Herceg and Ball [2] discuss extensively. For example, a search engine must extract text correctly from a document for that document to be queried and found by a user. A machine translation engine must extract text correctly from a document for it to generate an accurate translation of the document in another language. A text extractor, or text extraction toolkit, is the component that converts a file's textual content into a normalized text rendition that can be used by the aforementioned applications. However, developers often naïvely trust the accuracy of these tools and use them, without validation, in their file processing pipelines. Accuracy is only one part of performance; another is the ability of a tool to operate automatically on batches of files without human intervention. Text extractors are famous for improperly converting files, producing unusable garbled text, or mojibake [3]. Even if developers endeavored to validate a text extraction toolkit, such validation is challenging and labor intensive, and the industry provides no evaluation method or tools. This paper discusses an exploratory step toward filling this industry gap. This paper presents a method and a set of tools for evaluating a leading open source text extraction toolkit called Apache Tika (http://tika.apache.org/). Secondarily, this paper provides results from applying the method to compare the performance of Tika 1.5 and a prerelease version of Tika 1.7. Furthermore, we recommend actions for follow-on work so that the method and set of tools can be streamlined and made more useful.

2. Method

In this section we discuss a few methods for evaluating text extraction toolkits, including relevant software and corpus resources. Then we explain the particular method we explored in order to compare the performance of Tika 1.5 with a prerelease version of Tika 1.7. As we will show, the method we selected revealed several issues that we logged with an open source project—issues that would have otherwise been undetected for some time.

2.1. Text Extraction Tools

Wrapping a text extraction toolkit so that it can operate on a batch of files is essential for evaluating the performance of a given toolkit on many files. Tika provides no such wrapper. So, we developed one, called tika-batch. In addition, we designed this wrapper to be resistant to Tika failures (e.g., encountering problematic files), and provided a structured output format (i.e., JSON file with fields for

metadata and document content in primary files and attachments). The value of this wrapper extends well beyond an evaluation of Tika. This wrapper can be used directly in a file processing pipeline, for example, in the search engine pipeline discussed in Herceg, Allison, and Ball [4].

Next, we developed a comparison tool that generates a number of statistics that reveal differences in text extraction toolkit output. Consider the following scenario. A system administrator maintains a search engine that uses Tika to batch-convert a set of files into text-only renditions (e.g., using Tika 1.5). The system administrator observes that Apache has released a new version of the Tika toolkit (e.g., Tika 1.7). The system administrator is faced with the decision to upgrade, or keep running the installed version. Herceg [5] discusses the importance of evaluation for these kinds of technology decisions.

The following sections discuss how we used development versions of tika-batch and the comparison tool to evaluate Tika 1.5 versus a prerelease version of Tika 1.7, which we call Tika 1.7-SNAPSHOT.

2.2. Data

As a first step, we chose to use the publicly available *govdocs1* corpus (Garfinkel et al. [6]). There are nearly 1 million files in this corpus. After removing the files that the creators of the corpus identified as containing malware, there were 985,172 files, comprising roughly 470GB of data when unzipped. The creators of this corpus gathered these files from web servers in the .gov domain in 2009. The files include a range of formats. Table 1 shows the top 10 most common file extensions.

Table 1. Top 10 file extensions in the *govdocs1* corpus

File extension	Number of documents
pdf	231,009
html	214,264
jpg	109,094
txt	78,178
doc	76,507
xls	62,577
ppt	49,600
gif	36,279
xml	33,451
ps	22,012

It should be noted that this corpus is showing its age. The corpus contains only a limited quantity of the more recent Microsoft Office formats: 215 PPTX, 163 DOCX and 37 XLSX files. However, *govdocs1* is an invaluable resource that allows developers and researchers across the world to collaborate on an open corpus. Readers are encouraged to perform their own evaluations with a representative set of their own local data.

Given the source of the corpus, it is not surprising that the corpus is comprised mostly of English documents. In Table 2, we present the results from running a popular language identification package (https://code.google.com/p/language-detection/).

Table 2. Top 10 languages automatically identified in the *govdocs1* corpus

Language Code	Number of Documents Identified as that Language by Language-Detection Package
English	737,182
German	8,157
French	3,856
Spanish	3,822
Albanian	2,038
Italian	1,233
Portuguese	989
Vietnamese	908
Polish	852
Somali	673

2.3. Hardware, Software, and Configuration

For the evaluation, the authors used a 64-bit, 8 CPU virtual machine with 8GB of RAM. The operating system was Red Hat Enterprise Linux Server release 6.5 (Santiago), running Java 1.7.0_40-b43. For each run of a Tika version, we configured tika-batch to run with 10 file processors (10 threads).

There are a number of methods and measures that can be applied to evaluate the performance of a text extraction toolkit. The differing methods vary in the type of insight offered and in the cost to run the evaluation. Some common methods of evaluation include:

1) Functional Tests – run the extractor against a corpus and count:
   a) The estimated "single-thread time" – the sum of the times taken to process every document (if two documents were processed in two threads, and each took one second, the "single-

thread time" would be two seconds),
   b) The elapsed real time – the time it took from beginning to end according to a clock (if two documents were processed in two threads, and each took one second, the total clock time would be one second),
   c) The number of permanent hangs – how many documents triggered the text extractor to enter an apparently permanent hang,
   d) The number of fatal errors – how many documents triggered an error that required shutting down the process and restarting,
   e) The number of caught exceptions – how many documents caused the text extractor to throw an exception.
2) Comparison of Output with a Truth Set (i.e., ground truth) – manually extract content from a corpus and build a truth set, or an example for each file of what the text extraction toolkit should generate. There are various methods that one can use to compare the "truth file" to the extracted file.
3) Post-hoc Analysis – select a random sample of extracted documents and manually review the output.
4) Comparative Analysis of Functional Tests, Exceptions, Attachments, and Content – automatically compare the execution success and output of one tool with that of another. Using the same methods used for the truth set evaluation, identify:
   a) Comparison of the aforementioned functional tests,
   b) The number of "new exceptions" – files which threw an exception in the more recent version of the extractor that did not cause an exception in the earlier version of the extractor,
   c) The number of files with fewer attachments in the more recent version of the extractor than in the earlier version,
   d) The number of files that have substantive differences in their extracted text between one tool and another.

Although time and resource constraints limited the breadth of methods we could explore on this project, it was reasonable for us to select methods that applied to the use case presented in Section 2.1. As a result we chose to implement a comparative analysis of functional tests, exceptions, attachments, and content (4).

For item (d) of the comparative analysis, we determined that it would be unreasonable for a human to review the huge quantity of file differences that would occur. Therefore, we developed a heuristic filter that would pinpoint the Tika 1.5 and Tika 1.7-SNAPSHOT document pairs that contained substantive differences in extracted content. From this subset, one could select individual files for human review (e.g., a random sample).

The heuristic filter identified document pairs that had the highest probability of containing substantive differences. A key part of this filter was the use of a simple measure called the *Dice coefficient* (Manning and Schütze [7]). Specifically, we used the *Dice coefficient* between the content extracted from Tika 1.5 and the content extracted from Tika 1.7-SNAPSHOT. For tokenization, we relied on an Apache Lucene analyzer that included the ICUTokenizer and the ICUFoldingFilter.

Let us say that Tika 1.5 extracted "a b b c c d d e" from a given file and Tika 1.7-SNAPSHOT extracted "a b c d f". Borrowing technical terms from the field of corpus linguistics, we would say that the text extracted by Tika 1.5 had 8 "tokens" and 5 "unique tokens" (i.e., *types*). Tika 1.5's text has the following unique tokens: "a b c d e", and Tika 1.7-SNAPSHOT has "a b c d f". We calculated the *Dice coefficient* as 2 times the number of shared unique tokens divided by the sum of the unique tokens in both strings. In this case it would be 2*4/10, or 0.80. The *Dice coefficient* scales the similarity score between 0.0 and 1.0, with 1.0 being perfect similarity.

Dice coefficient applies to this evaluation because it approximates an answer to the question of: "if I were to search for a single term in a document, would that document be retrieved if I used one version of the text extraction toolkit versus the other?" The *Dice coefficient* focuses on the presence or absence of terms in a document (would the document be retrieved or not for a given term) versus other measures that focus on "token" overlap.

We chose to concatenate the content text from embedded documents with the extracted content text from each main document. Therefore, if the earlier version of Tika was not extracting as many embedded documents as the later, we might expect to find a *Dice coefficient* of less than 1.0.

We applied the *Dice coefficient* in the heuristic filter that strategically identified the Tika 1.5 and Tika 1.7-SNAPSHOT document pairs with the highest probability of containing substantive differences. Specifically, we selected Tika 1.5 and Tika 1.7-SNAPSHOT document pairs that had all of the following criteria:

1) The same number of attachments
2) More than 30 unique tokens in either of the documents
3) Less than a 0.90 *Dice coefficient*, or the documents differed in more than 100 unique tokens

We selected the above criteria for the following reasons. We chose the first condition because we would expect a lower *Dice coefficient* if two documents had a different number of attachments. We chose the second condition because we are not interested in documents with only a few unique tokens; and documents with only a few unique tokens might have inflated differences in *Dice coefficients* (for example, if each document in a pair only has two unique tokens and they only have one unique token in common, the *Dice coefficient* will be 0.33). We chose the final condition to identify documents where there may be an important difference between the text extracted by Tika 1.5 versus Tika 1.7-SNAPSHOT.

We anticipated that evaluating on such a large data set would yield too many files to manually review. Therefore, we further selected a very small subset of document pairs for manual inspection. This human review revealed the fine-grained differences between Tika 1.5 and Tika 1.7-SNAPSHOT.

3. Results

In this section we present the results from applying our exploratory text extraction toolkit evaluation method.

The total estimated "single-thread time" for processing the corpus was 42 hours for Tika 1.5 and nearly 36 hours for Tika 1.7-SNAPSHOT. The elapsed real time was 5 hours for Tika 1.5 and slightly less than 4.5 hours for Tika 1.7-SNAPSHOT.

There were 8 out-of-memory errors for Tika 1.5, and 5 files that caused permanent hangs. For Tika 1.7-SNAPSHOT, there were 6 files that caused out-of-memory errors and 6 that caused permanent hangs. Note that the number of out-of-memory errors vary depending on the run; if, during one run, two large files are processed at the same time, together they might cause an error, whereas, if they were not processed simultaneously, there might not be an out-of-memory error.

Table 3 shows the number of exceptions that the comparison tool found when trying to read the output of the extraction process. This can happen if there is a zero byte file or if a writer/parser was interrupted while writing the JSON file. Note that there were conspicuously more PDF exceptions for Tika 1.7-SNAPSHOT, but there were fewer exceptions for all other extensions.

Table 3. JSON exceptions by file extension

File extension	Tika 1.5	Tika 1.7-SNAPSHOT
pdf	44	57
ppt	10	4
txt	10	6
doc	2	6
kmz	2	0
xls	2	0
csv	1	2
html	1	1
pps	1	1
ps	0	1
unk	0	1

Table 4. Caught exceptions, Tika 1.5 vs. Tika 1.7-SNAPSHOT by file extension

File extension	Tika 1.5	Tika 1.7-SNAPSHOT
xls	2,824	2,828
log	1,253	1,253
ppt	2,195	1,191
doc	847	795
pdf	644	123
xml	417	417
html	161	161
pps	28	8
unk	20	18
kml	19	19
txt	8	6
jpg	5	5
pptx	3	3
rtf	3	2
tmp	2	2
text	2	0
docx	1	1
sgml	1	1
NO_SUFFIX	1	1
TOTAL	8,434	6,834

Table 4 shows the number of caught exceptions for the two versions of Tika. For Tika 1.7-SNAPSHOT, there was a noticeable reduction in caught exceptions

in PPT files and PDF files.

We found an abundance of exceptions related to Tika's XML parser, which only processes compliant XML. If a file contains non-compliant XML, the parser throws an exception and does not return any text. Nearly all of the exceptions for the following file extensions were actually XML parse exceptions: LOG, XML, KML, and HTML. For Tika 1.7-SNAPSHOT, these XML exceptions accounted for more than 25% of all of the observed exceptions.

We were also interested in identifying files that had exceptions in Tika 1.7-SNAPSHOT but did not have exceptions in Tika 1.5 (i.e., new exceptions). Table 5 shows that new exceptions occurred at a very low frequency.

Documents of various formats can have embedded documents. Typical container files include ZIP files or TAR files, but files can also be embedded in RTF, PDF, DOC, PPT, XLS and other Microsoft Office formats. We found that there was only one DOC file that had fewer attachments with Tika 1.7-SNAPSHOT than with Tika 1.5. However, there were several files that had more attachments with Tika 1.7-SNAPSHOT than with Tika 1.5, as shown in Table 6. In other words, Tika 1.7-SNAPSHOT was more successful than Tika 1.5 at extracting text from attachments.

Table 5. New exceptions in Tika 1.7-SNAPSHOT

File extension	Number of files with exceptions in Tika 1.7-SNAPSHOT but not in Tika 1.5
xls	6
ppt	4
doc	2
pdf	2
xml	1

Table 6. Number of files with more embedded documents extracted in Tika 1.7-SNAPSHOT than in Tika 1.5

File extension	Number of files
pptx	188
rtf	182
pdf	30
docx	10
doc	10
xlsx	8
zip	4
text	1

According to our method, we applied a heuristic filter to identify and count the number of files that showed substantive differences in the extracted content. Table 7 shows the number of files that have the following criteria: the same number of attachments, greater than 30 unique tokens in either of the documents, and a similarity score of less than 0.90 or a unique token count that diverges by more than 100.

Table 7. Files that met the heuristic filter for potential manual review.

File extension	Number of files
pdf	618
xls	101
java	95
html	6
gz	4
doc	1

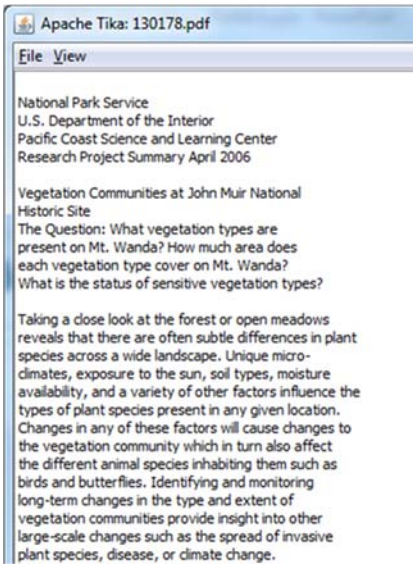

Figure 1. Text extracted with Tika 1.5 from 130178.pdf

We selected a few of these documents for detailed inspection, in order to determine if the content was better or worse with Tika 1.7-SNAPSHOT. During manual review of the files identified by the heuristic, we found differences that indicated new problems with Tika 1.7-SNAPSHOT's PDF parser (i.e., regressions in Apache PDFBox). The version of PDFBox used in Tika 1.7-SNAPSHOT appeared to regress in at least two ways from the version of PDFBox used in Tika 1.5. First, some files were truncated with Tika 1.7-

SNAPSHOT. Second, Tika 1.7-SNAPSHOT appeared to regress in character mapping. Figures 1 and 2 depict an example. Figure 1 shows the text extracted from 130178.pdf with Tika 1.5, and Figure 2 shows the text extracted with Tika 1.7-SNAPSHOT. Notice that Figure 2 is full of mojibake.

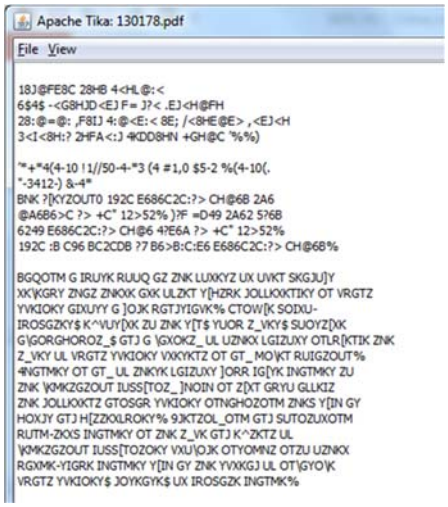

Figure 2. Text extracted with Tika 1.7-SNAPSHOT from 130178.pdf

This detailed inspection had an immediate positive impact on the Apache Tika and PDFBox projects. We opened issues with the PDFBox project for each of the aforementioned findings: PDFBox-2376 and PDFBox-2377. We also opened issue TIKA-1419. These published issues and ongoing collaboration using the *govdocs1* corpus allowed a committer on the PDFBox project to discover other causes for regression and to make fixes to PDFBox. These contributed to an upgrade to PDFBox 1.8.8, which is tracked on TIKA-1442. The details of these posted issues can be searched at https://issues.apache.org/jira/i#browse.

One hundred one XLS document pairs had low *Dice coefficient* scores. A review of three of these XLS files showed an improvement in Tika 1.7 SNAPSHOT's Microsoft Office parser (Apache POI) in comparison with Tika 1.5. Specifically, Tika 1.5's parser (the older version of the Microsoft Office parser), was incorrectly adding underscores to some numbers.

Further review of files with low *Dice coefficient* scores led us to compressed GZ files. Manual review of one of these files revealed that, again, the low score was due to regressions in Tika 1.7 SNAPSHOT's PDF parser.

Still further review of low scoring document pairs revealed a file type identification improvement in Tika 1.7 SNAPSHOT. We reviewed one of the HTML files (487828.html), and found that it was an FDF file (PDF font descriptor file), not an HTML file. Tika 1.7-SNAPSHOT correctly identified it as an FDF file, whereas Tika 1.5 identified it as an HTML file.

To summarize, using the evaluation method we found that Tika 1.7-SNAPSHOT regressed in PDF parsing ability, improved in XLS parsing ability, and threw fewer exceptions for both PDF files and PPT files. Furthermore, this evaluation method revealed that the strict Tika XML parser is problematic. Users of Tika might consider alternatives to this parser (e.g., the more lenient Tika HTML parser).

4. Summary

Text extraction toolkits are vital components of a number of popular software applications. Developers need objective measures of a toolkit's performance before deciding to embed it in their larger software application. Therefore, text extraction toolkit evaluation tools and methods must be developed. In this paper we discussed some common methods to evaluate text extraction toolkits, and presented the specific method that we used to evaluate Tika 1.5 versus a prerelease version of Tika 1.7. The method involved a comparative analysis of functional tests, exceptions, attachments, and content. Furthermore, we used a heuristic filter in order to strategically identify the subset of output document pairs that had a high probability of containing substantive differences. This filtering allows the human review to be focused on the most revealing document pairs.

The application of this evaluation method revealed several regressions in Tika's PDF parser, and we were able to bring these issues to the attention of the PDFBox project team. These postings and follow-on collaboration using the *govdocs1* corpus resulted in a positive ripple effect on the open source project, with PDFBox developers identifying and posting more issues. This particular benefit to an open source project indicates the usefulness of the evaluation method for text extraction toolkit regression testing. Overall for the *govdocs1* corpus, Tika 1.7-SNAPSHOT had fewer exceptions than Tika 1.5, especially with PDF files and with PPT files. If the fixes are made in PDFBox before Tika 1.7 is released, then, overall, the changes in Tika 1.7 versus Tika 1.5 would be largely positive, at least for the *govdocs1* corpus.

The results of this study apply to the *govdocs1* corpus. As noted, this corpus is aging, and nearly all of the documents are in English. The authors strongly encourage readers to carry out the above types of evaluation on a representative sample of their documents to identify potential limitations in the text extraction component or to identify strengths and weaknesses when comparing two text extraction toolkits or two different versions of the same text extraction component.

We believe that the evaluation method outlined here can be built into a new component for Tika. Such a component can follow the model of another popular open source project: Solr. Solr's administration user interface provides several features, including the evaluation of queries and indexes. The authors plan to develop and contribute a similar tool, tika-eval, that will enable users to carry out evaluations of Tika and other text extraction toolkits.

5. Future Work

During work on this exploratory evaluation effort, we identified actions that would improve the overall method in both the short term and in the longer term. In the short term we plan to pursue developing a publicly available text extraction evaluation toolkit (e.g., tika-eval). In the toolkit we plan to include automatic calculation and reporting of functional test and comparison statistics. Also, a simple user interface would help developers strategically select and review document pairs with a high probability of substantive differences. Recall that the process of determining whether differences are good or bad is labor intensive, involving manual review of file content (i.e., document pairs). This toolkit will be instrumental in guiding the review of document pairs with substantive differences. Additionally it is targeted to enable random sampling after the heuristic filter is applied, and provide a calculation of confidence intervals.

A single metric to identify text extraction failures is elusive and requires research. Longer term plans include investigating metrics for automatically identifying when a text extractor has failed to extract useful text from a document. Ideally, these metrics would be language and document format agnostic. Developers could use these metrics to improve parsers, and integrators could use these metrics in thresholds to determine whether or not to process a document.

It is our hope that the work described here will (a) encourage the continued development of corpora similar to that of *govdocs1*, preferably publicly available, in order to leverage the effort of open source developers, and (b) encourage Tika users to actively submit trouble reports and patches to the Tika project.

References

1. Herceg, P. M. The content extraction technology gap: Accessing the textual content of computer files. Technical Report MTR090437. McLean, VA: The MITRE Corporation. December 7, 2009.

2. Herceg, P. M. & Ball, C. N. Reliable electronic text: The elusive prerequisite for a host of human language technologies. Technical Report MTR100302. McLean, VA: The MITRE Corporation. September 30, 2010. Retrieved 10/17/14 from http://www.mitre.org/sites/default/files/pdf/11_0690.pdf.

3. Mojibake. In *Wikipedia*. Retrieved 10/17/14 from http://en.wikipedia.org/wiki/Mojibake.

4. Herceg, P. M., Allison, T. B., & Ball, C. N. A MITRE search prototype using Outside In and Lucene. Technical Report MTR090085. McLean, VA: The MITRE Corporation. March 30, 2009.

5. Herceg, P. M. Defining useful technology evaluations. Technical Report MTR070061R1. McLean, VA: The MITRE Corporation. September 13, 2007. Retrieved 10/17/14 from http://www.mitre.org/sites/default/files/pdf/08_0038.pdf.

6. Garfinkel, S., Farrell, P., Roussev, V., & Dinolt, G. Bringing science to digital forensics with standardized forensic corpora. *Digital Investigation*, 6, S2-S11. 2009. Retrieved 10/20/14 from http://www.sciencedirect.com/science/article/pii/S1742287609000346.

7. Manning, C., & Schütze, H. Foundations of statistical natural language processing. Cambridge, MA: The MIT Press. 1999.