ComparisonTikaAndPDFToText201811

Background

After refreshing Tika's regression corpus (see CommonCrawl3 and TIKA-2750), we thought it might be interesting to run a comparison between the text extracted with pdftotext and Tika/PDFBox. Given that pdftotext does not extract content from embedded files and given that it does not perform Optical Character Recognition (OCR) or offer integration with OCR, this evaluation focused only on extracting electronic text as stored within PDFs. The goals of this study include:

1. Identify areas for improvements for PDFBox and pdftotext 2. Identify areas for improvements for tika-eval

The reader should **not** read the following as a recommendation for one tool over another.

The tika-eval reports and the full H2 database of comparison results are available here: http://162.242.228.174/pdf_parsing/pdftotextVPDFBox_201811/.

NOTE: On December 11, 2018, I reran the analysis with the pdftotext commandline option *-cfg* to specify the (font) configuration file. During the first run and the initial analysis, *pdftotext* was not applying the custom fonts even though the xpdfrc file was placed in */usr/local/etc/xpdfrc*. This wiki has been updated to reflect the slight differences in the re-run of the evaluation. In general, it appears, based on the common words metric, that text extraction did improve for Japanese and Chinese PDFs when we used the *-cfg* option to manually specify the *xpdfrc* config file that points to the language-specific extra font files.

Tools and Data

- Operating system: Linux cloud-server-02 3.10.0-327.10.1.el7.x86_64 #1 SMP Sat Jan 23 04:54:55 EST 2016 x86_64 GNU/Linux (Red Hat Enterprise Linux Server release 7.0 (Maipo)). We relied on the default system fonts. We have made no modifications to the default OS, nor have we installed fonts.
- pdftotext – we downloaded the most recent available binaries, version 4.00.01, and we followed the directions to install all language modules (see virtual machine). We wrote a simple Groovy wrapper to call a new pdftotext process for every file; if no extract file was generated by pdftotext, the Groovy script generated a 0-byte file; also, we forced a timeout after 300 seconds (5 minutes).
- Tika/PDFBox – we used a snapshot version of Tika 1.20, which uses PDFBox 2.0.12. We used the default settings and did not sort by position, etc. We did enable permissions checking so that text was not extracted from PDF files that did not allow text extraction.
- Tika identified 528,618 PDF files in the new pull from Common Crawl. Many of these files are truncated, and 6,787 caused permission exceptions (these are either encrypted or they do not allow extraction of text).

Exceptions

There were 58,077 empty files generated by our wrapper of pdftotext. Given that pdftotext respects access permissions, that means that there may have been up to 51,290 runtime exceptions; of these, seven files caused timeouts. Some kind of content (number of tokens > 0) was extracted for 400,782 files; there were 373,569 extracts that contained >= 100 tokens.

Aside from the "permission exceptions", there were 38,158 files that caused a runtime exception for PDFBox. Some kind of content (number of tokens > 0) was extracted for 428,706 files. Note that for this evaluation, we used a standard default content handler that appends the title to the extracted content. Therefore, we also report that there were 384,277 extracts that contained >= 100 tokens.

High Level Comparison

In the absence of ground truth, tika-eval counts the number of "common words" (see: Common Words) per file as a rough proxy for extraction quality. The extracts generated by pdftotext contained 612,693,477 "common words", and the extracts from Tika/PDFBox contained 615,269,581. By this metric, that would be a 0.42% increase in the extraction of common tokens if one moved from pdftotext to Tika/PDFBox.

In manually reviewing the results, we noticed that Tika/PDFBox was apparently performing far better when there was a disagreement in language id (see the section below on Languages). We noticed that as the content between the two extracts became more similar, there were often more examples of areas for improvement in spacing in Tika/PDFBox's extracts than in pdftotext's extracts. Based on this observation, we calculated the number of common tokens extracted for each tool when the language id for both extracts was the same *or* where there was only content for one of the extracts (but not the other) [SQL1] and [SQL2]. When we do this, the difference narrows even more – pdftotext extracts 607,187,545 common tokens, whereas PDFBox/Tika extracts 607,324,140 (a 0.02% increase).

Bottom line: based on this analysis so far, the differences between the tools for extracting electronic text are not great. It appears that PDFBox extracts slightly more content. However, the reports and this preliminary analysis point to areas for improvement in both tools and in tika-eval.

Languages

In the following, we show the top 20 languages identified in the extracted text. The tika-eval module uses the optimaize language detector (version 0.6) for language identification. The language codes are roughly ISO 639-1.

The first language is that identified in the extract from pdftotext, and the second is the language identified on the extract of PDFBox. For example 'en->fa' means that language id returned 'en' on the pdftotext extract, but 'fa' on the Tika/PDFBox extract.

Language id	Number of Files
en->en	143,784
ru->ru	44,460
fr->fr	38,872
it->it	36,433
de->de	30,151
es->es	18,335
ja->ja	16,106
el->el	9,761
fa->fa	8,486
ko->ko	8,213
zh-cn->zh-cn	5,815
tr->tr	5,477
null	3,132
vi->vi	2,981
he->he	2,280
ar->ar	2,087
ca->ca	1,275
en->fa	1,240
pt->pt	1,105
de->en	860

In the following, we show the top 10 language id pairs, where the language id differs between the extracts.

Language ids	Number of Files
en->fa	1,240
de->en	860
en->de	519
en->bn	392
ar->fa	391
it->en	209
en->it	209
fr->en	201
en->fr	149
br->bn	146

"Common Words" per Language (id)

In the following table, we present the number of "common words" extracted per language id [SQL3]. Note that this language "id" is based on the extracted text per tool, *not* ground truth. Some care has to be made in interpreting this data.

Language Id	pdftotext	Tika /PDFBox	% Change
null	5,729	2,757	-51.9%
af	10,513	11,192	6.5%
an	113,326	88,633	-21.8%
ar	2,272,891	1,832,365	-19.4%
ast	15,507	16,927	9.2%
be	34	74	117.6%
bg	22,582	25,887	14.6%
bn	5,648	201,707	3471.3%

br	21,470	21,282	-0.9%
ca	301,284	307,474	2.1%
cs	26,933	32,126	19.3%
cy	72,368	70,987	-1.9%
da	24,702	29,576	19.7%
de	30,132,564	31,337,406	4.0%
el	9,432,310	9,572,594	1.5%
en	245,298,469	256,492,056	4.6%
es	39,476,601	40,478,683	2.5%
et	38,174	23,226	-39.2%
eu	17,771	14,476	-18.5%
fa	19,376,868	21,751,814	12.3%
fi	10,195	10,628	4.2%
fr	61,980,431	65,709,586	6.0%
ga	28,809	25,165	-12.6%
gl	206,752	220,055	6.4%
gu	3,456	3,840	11.1%
he	3,389,105	3,358,894	-0.9%
hi	266,969	264,400	-1.0%
hr	30,780	30,632	-0.5%
ht	8,498	3,570	-58.0%
hu	13,283	15,661	17.9%
id	215,824	193,661	-10.3%
is	14,480	13,250	-8.5%
it	45,985,473	47,562,781	3.4%
ja	45,187,665	46,921,042	3.8%
km	5	9	80.0%
kn	4,090	4,285	4.8%
ko	4,908,528	4,990,836	1.7%
lt	5,766	5,822	1.0%
lv	12,467	10,453	-16.2%
mk	547	1,400	155.9%
ml	1,281	1,280	-0.1%
mr	22,775	23,452	3.0%
ms	234,536	286,100	22.0%
mt	39,416	29,780	-24.4%
ne	76	234	207.9%
nl	563,761	584,303	3.6%
no	55,053	56,548	2.7%
oc	838	5,017	498.7%
pa	79	107	35.4%
pl	51,976	55,890	7.5%
pt	2,196,276	2,395,862	9.1%
ro	40,358	32,007	-20.7%
ru	79,309,911	81,574,518	2.9%
sk	9,131	7,760	-15.0%
sl	9,243	13,665	47.8%
so	281,724	350,402	24.4%
sq	3,089	5,997	94.1%
sr	690	707	2.5%
sv	56,956	69,112	21.3%
sw	1,180	1,003	-15.0%
ta	1,308	1,303	-0.4%
te	3,360	5,824	73.3%
th	5,673	8,675	52.9%

tl	1,277	2,987	133.9%
tr	878,127	917,616	4.5%
uk	3,977	5,910	48.6%
ur	29,711	9,575	-67.8%
vi	2,310,556	2,455,051	6.3%
yi	28	32	14.3%
zh-cn	16,938,972	18,231,035	7.6%
zh-tw	703,272	646,268	-8.1%

When we require that both extracts for a given file have the same language id, we see some different patterns [SQL4].

Language Id	pdftotext	Tika /PDFBox	% Change
af	10,274	10,330	0.5%
an	68,416	63,324	-7.4%
ar	2,021,212	1,652,127	-18.3%
ast	9,931	10,009	0.8%
be	33	34	3.0%
bg	20,522	20,549	0.1%
bn	4,556	4,754	4.3%
br	7,207	7,340	1.8%
ca	242,815	246,346	1.5%
cs	24,872	25,297	1.7%
cy	51,644	51,330	-0.6%
da	22,737	23,412	3.0%
de	29,478,164	29,814,286	1.1%
el	9,427,793	9,371,790	-0.6%
en	243,539,028	244,571,204	0.4%
es	38,876,299	38,835,666	-0.1%
et	16,080	16,516	2.7%
eu	11,018	11,127	1.0%
fa	19,278,448	17,596,803	-8.7%
fi	9,767	9,913	1.5%
fr	61,593,227	62,039,806	0.7%
ga	22,256	22,131	-0.6%
gl	174,194	166,007	-4.7%
gu	3,456	3,633	5.1%
he	3,378,638	3,206,568	-5.1%
hi	265,152	242,522	-8.5%
hr	30,124	30,354	0.8%
ht	2,798	2,869	2.5%
hu	13,260	13,595	2.5%
id	190,295	190,072	-0.1%
is	10,718	10,777	0.6%
it	45,675,112	45,575,218	-0.2%
ja	45,046,240	45,903,641	1.9%
km	5	9	80.0%
kn	3,938	3,950	0.3%
ko	4,864,390	4,914,251	1.0%
lt	5,587	5,684	1.7%
lv	10,786	10,435	-3.3%
mk	545	1,398	156.5%
ml	1,281	1,280	-0.1%
mr	22,695	22,523	-0.8%
ms	221,213	226,932	2.6%
mt	18,253	18,777	2.9%
ne	73	83	13.7%

nl	548,538	552,625	0.7%
no	41,588	42,482	2.1%
oc	605	609	0.7%
pa	79	107	35.4%
pl	50,848	51,776	1.8%
pt	2,090,127	2,144,491	2.6%
ro	30,282	30,889	2.0%
ru	79,195,319	78,271,782	-1.2%
sk	8,745	6,776	-22.5%
sl	8,290	8,519	2.8%
so	224,340	212,438	-5.3%
sq	2,882	4,269	48.1%
sr	689	703	2.0%
sv	40,347	41,313	2.4%
sw	959	951	-0.8%
ta	1,308	1,303	-0.4%
te	3,360	3,407	1.4%
th	5,078	5,088	0.2%
tl	1,175	1,207	2.7%
tr	865,494	878,297	1.5%
uk	3,898	5,153	32.2%
ur	21,459	5,553	-74.1%
vi	2,254,738	2,264,384	0.4%
yi	28	32	14.3%
zh-cn	16,608,321	17,263,436	3.9%
zh-tw	347,067	349,029	0.6%

Further evaluation and analysis are required, but we should look into:

1. Why there are so many "common words" for *bn* in the first common tokens by language table? 2. Are there systematic areas for improvements in PDFBox for *hi* (-8.5%), *he* (-5.1%) and Arabic script languages: *ar* (-18%), *fa* (-9%), *ur* (-74%)?

Most importantly, we need to determine if any of the above areas for inquiry are based on faults in tika-eval that should be fixed.

Follow up Analysis

1. Why there are so many "common words" for *bn* in the first common tokens by language table?

I ran [SQL5], and I manually reviewed results. I observed the following points:

1. There was only one out of the 100 documents that had what looked like Bangla words in the top 10 most common words for those documents. 2. My intuition from previous experience with Optimaize, and it was confirmed in looking at the top 10 words for these documents is that Optimaize prefers *bn* when there are many numerals and very little other language content. 3. As in previous work with Optimaize, I was struck that the confidence levels are typically very high (~0.999) even when there is very little content. For example, *commoncrawl3/7A/7AZUB5NHLJN3TBCMEP2YRSRK6DDNBP5F* is mostly comprised of the UTF-8 replacement character, "EF BF BD" (equivalent to U+FFFD) (~13,000 of these); there are a few new lines, a few tabs, a few numerals, and the word 'untitled', and yet Optimaize's confidence is *0.9999907612800598* that this is Bangla. 4. The current OOV% metric does not take calculate a confidence. If there's just one alphanumeric term and it happens to be in the dictionary, then the OOV% is 0%, which is less than entirely useful. It would be better to improve our "language-y" score or its inverse, the "junk" score (see TIKA-1443), to include a confidence interval based on the amount of input. 5. When tika-eval doesn't have a "common words" list for a language, e.g., *bn*, it backs-off and uses the English list. Given that the internet is overwhelmingly English and given that the *commoncrawl3* regression corpus contains quite a bit of English, and given that content from the title metadata field slipped into the extracted text for the PDFBox/Tika extracts, this backing-off to English can lead to misleading results.

My conclusion is that most of the documents that received a language id of *bn* actually contain a high percentage of junk.

Recommendations:

1. We should experiment with other language detectors and evaluate them for the traditional language-id performance measures: accuracy and speed on known language content. However, we should also evaluate them on how well they handle various types of degraded text to confirm that the confidence scores are related to the noise – content that contains 98% junk text should not receive a language id confidence of 99.999%. 2. We should augment our "common words" lists to cover all languages identified by whichever language detector we choose. We should not back-off to the English list for "common words". 3. We should continue to work on/develop a junk metric that is more nuanced than the simple sum of "Common Tokens" and the OOV%. The metrics should take the following into account:

a. Amount of evidence. 2. Alignment of distribution of token lengths relative to the "id'd" language (this will be useless with CJK, which are simply bigrammed by tika-eval; but it might be very useful for most other languages). 3. Amount of symbols and U+FFFD characters vs. the alphabetic tokens. 4. Instead of binary OOV%, it might be useful to calculate alignment to a Zipf distribution or simply similarity to a language model – we'd need to include % of words in the common words file. 5. Incorrect duplication of text. For file, *commoncrawl3/2E/2EXCWC7T6P5ZY6DINFI3X2UQNIMAISKT*, tika-eval shows an increase in Common Tokens of 50,372 tokens if switching from pdftotext to PDFBox/Tika. However, this file has an absurd amount of duplicate text in the headers – 17,000 occurrences of "training" in the PDFBox/Tika extract, and only 230 in the pdftotext extract. PDFBox/Tika correctly suppresses these duplicate text portions if *setSuppressDuplicateOverlappingText* is set to *true*, but Tika's default is not to suppress duplicate text. One consideration is that for this file, the % of OOV is 39% in pdftotext but only 8% in the text extracted by PDFBox/Tika. This suggests that it might be better, instead of simply summing the common tokens, to sum them only in files which have an OOV% which is within the norm (say, one stddev). As a side note, 40% is fairly common for OOV for English documents – the median is 45%, and the stddev is 14%.

2. Are there systematic areas for improvements in PDFBox for *hi* (-8.5%), *he* (-5.1%) and Arabic script languages: *ar* (-18%), *fa* (-8%), *ur* (-74%)?

I don't know these languages, but I ran [SQL7] and then put the contents of *TOP_10_UNIQUE_TOKEN_DIFFS_A* and *TOP_10_UNIQUE_TOKEN_DIFFS_* through Google translate. For example, for the top 10 unique words in *commoncrawl3_refetched/XH/XHYIWIBT5QPY64UYUPLXZXAYC2I5JPZS*:

```
: 532 | : 520 | : 450 | : 370 | : 365 | : 343 | : 342 | : 297 | : 280 | : 254
```

are translated as:

```
I: 532 | Is: 520 | Of: 450 | Are: 370 | Yes: 365 | Who: 343 | From: 342 | : 297 | We: 280 | Mr.: 254
```

Whereas PDFBox/Tika's unique tokens

```
: 564 | : 537 | : 468 | : 386 | : 365 | : 360 | : 348 | : 306 | : 281 | : 250
```

are translated as:

```
Th: 564 | Yes: 537 | S: 468 | Yes: 386 | Hak: 365 | Ki: 360 | S: 348 | A: 306 | Mah: 281 | Ingredients: 250
```

Overall, this method wasn't able to yield satisfactory insight about general patterns. In some cases, the individual terms looked better in one tool or the other and *vice versa*.

I did note that there were more cases in PDFBox's extracted text of numerals concatenated with words as in *commoncrawl3/JG/JGE6WTYI5SEI3Z4JUULIPSSRTNL3VMIG:*

TOP_10_UNIQUE_TOKEN_DIFFS_A

```
        1: 167 | : 167 | 9: 44 | 8: 38 | 7: 28 | 6: 16 | 5: 9 | 4: 6 | 3: 2 | 9622243
```

TOP_10_UNIQUE_TOKEN_DIFFS_B

```
        : 44 | 8: 38 | 7: 28 | 10: 24 | 6: 16 | 5: 9 | 4: 6 | 3: 2 | 96222431: 1
```

Post-Study Reflection/Areas for Improvements

Overall improvements to this process

- The wrapper around pdftotext should have "caught" the exception written to stderr and stored that as we do with exceptions from Tika.
- Tika currently includes the file's 'title' metadata in the content of the file. This gives the misleading impression that some content was extracted from the file when, in fact, only the title was extracted from the XMP or metadata. Next time, we should use a content handler that only includes the extracted text.
- Next time we run this evaluation, we should specify *-cfg* from the commandline and/or figure out why our *pdfrc* file wasn't being read where we placed it.

Improvements to tika-eval

- We observed a handful of cases where the number of "common words" increased, but the content extracted was probably worse between two tools. This happened when one tool added spaces incorrectly, but the sub-words were actual words within the language. See, for example: *comm*

oncrawl3/TF/TFNFGXL27M77Q6X42ECYWJNSJ32WES74 (Russian) and *commoncrawl3/FS/FSEHYPPOEV6EUYND5BRP3BBNAI5FVPYP* (German) ("fachgruppe" vs "fach gruppe" and "ermoglicht" and "ermog" "licht")
- If there's an "extract exception", meaning an empty file or an incomplete json file, we include that information in the containers table, but we don't include a row for that file in the profiles table. This causes some of the SQL that ships with tika-eval to result in not-quite-fair comparisons; some of the SQL that takes into account "runtime exceptions" fails to take into account "extract exceptions."
- See the point above about improving the "junk" metric.

SQL

[SQL1]

```
select sum(cb.num_common_tokens) from contents_b cb
join profiles_b pb on pb.id=cb.id
left join profiles_a pa on pb.id=pa.id
left join contents_a ca on pa.id=ca.id
where pa.is_embedded = false and pb.is_embedded=false
and (ca.lang_id_1 = cb.lang_id_1
or ca.lang_id_1 is null)
```

[SQL2]

```
select sum(ca.num_common_tokens) from contents_a ca
join profiles_a pa on pa.id=ca.id
left join profiles_b pb on pa.id=pb.id
left join contents_b cb on pb.id=cb.id
where pa.is_embedded = false and pb.is_embedded=false
and (cb.lang_id_1 = ca.lang_id_1
or cb.lang_id_1 is null)
```

[SQL3]

```
select lang_id_1, sum(num_common_tokens) as total_common_tokens
from contents_b
group by lang_id_1
order by lang_id_1
```

[SQL4]

```
select ca.lang_id_1, sum(ca.num_common_tokens)
from contents_a ca
join contents_b  cb on ca.id=cb.id
where ca.lang_id_1=cb.lang_id_1
group by ca.lang_id_1
order by ca.lang_id_1
```

[SQL5]

```
select ca.lang_id_1, ca.top_n_tokens, cb.top_n_tokens from contents_b cb
join contents_a ca on cb.id=ca.id
where cb.lang_id_1 = 'bn'
order by rand()
limit 100;
```

[SQL6]

```
select  ca.id, file_path,
1-(cast(ca.num_common_tokens as float) / cast(ca.num_alphabetic_tokens as float)) as OOV_A,
ca.num_alphabetic_tokens,
1-(cast(cb.num_common_tokens as float) / cast(cb.num_alphabetic_tokens as float)) as OOV_B,
cb.num_alphabetic_tokens,
ca.lang_id_1, ca.lang_id_prob_1,
cb.lang_id_1, cb.lang_id_prob_1,
```

```
ca.top_n_tokens, cb.top_n_tokens
from contents_b cb
join contents_a ca on cb.id=ca.id
join profiles_a pa on ca.id=pa.id
join containers c on pa.container_id=c.container_id
where cb.lang_id_1 = 'bn' and
ca.num_alphabetic_tokens > 0
and cb.num_alphabetic_tokens > 0
order by OOV_B asc
limit 100;
```

[SQL7]

```
select file_path, ca.top_n_tokens, cb.top_n_tokens,
(cb.num_common_tokens-ca.num_common_tokens) as delta_common_tokens,
top_10_unique_token_diffs_a, top_10_unique_token_diffs_b
from contents_a ca
join contents_b cb on ca.id=cb.id
join content_comparisons cc on cc.id=ca.id
join profiles_a pa on ca.id=pa.id
join containers cc on pa.container_id=cc.container_id
where ca.lang_id_1='ur'
and cb.lang_id_1='ur'
order by delta_common_tokens asc
```

How to make sense of the tika-eval reports

Exceptions aside, the critical file is *content/content_diffs_with_exceptions.xlsx*. This shows differences in the content that was extracted. Column *TOP_10_UNIQUE_TOKEN_DIFFS_A* records the top 10 most frequent tokens that appear only in "A" extracts (pdftotext); *TOP_10_UNIQUE_TOKEN_DIFFS_B* records the top 10 most frequent tokens that appear only in "B" extracts (Tika/PDFBox); *NUM_COMMON_TOKENS_DIFF_IN_B* records whether there has been an increase (positive number) or a decrease in "common tokens" if one were to move from "A" to "B" as the extraction tool.

For example, for file *commoncrawl3_refetched/7L/7L6BDSEYCY3QVVPM7YYK3FVCK2ZLOSA7*, *NUM_COMMON_TOKENS_DIFF_IN_B* has a value of '38', which suggests that there are 38 more "common words" in the text extracted by Tika/PDFBox than by pdftotext.

TOP_10_UNIQUE_TOKEN_DIFFS_A has

```
bklasse: 2 | gehoben: 2 | jahressiegers: 2 | untersagt: 2 | verlasslichkeit: 2 | 3321: 1 | 50jahrigen: 1 |
5282708: 1 | 60jahriges: 1 | 970843: 1
```

TOP_10_UNIQUE_TOKEN_DIFFS_B has

```
e: 23 | gen: 9 | be: 6 | te: 6 | schaft: 5 | gung: 4 | nant: 4 | nen: 4 | o: 4 | ten: 4
```

This probably means that some words were incorrectly split by PDFBox ("tool B"); and it may mean that a hyphen was incorrectly dropped in a few words by pdftotext ("tool A"): "bklasse", "50jahrigen" and "60jahriges", which should probably be "b-klasse", "50-jahrigen" and "60-jahriges"