

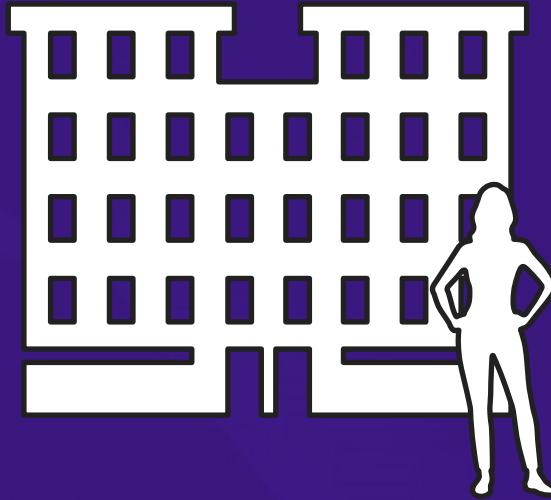
PDF and Open Source

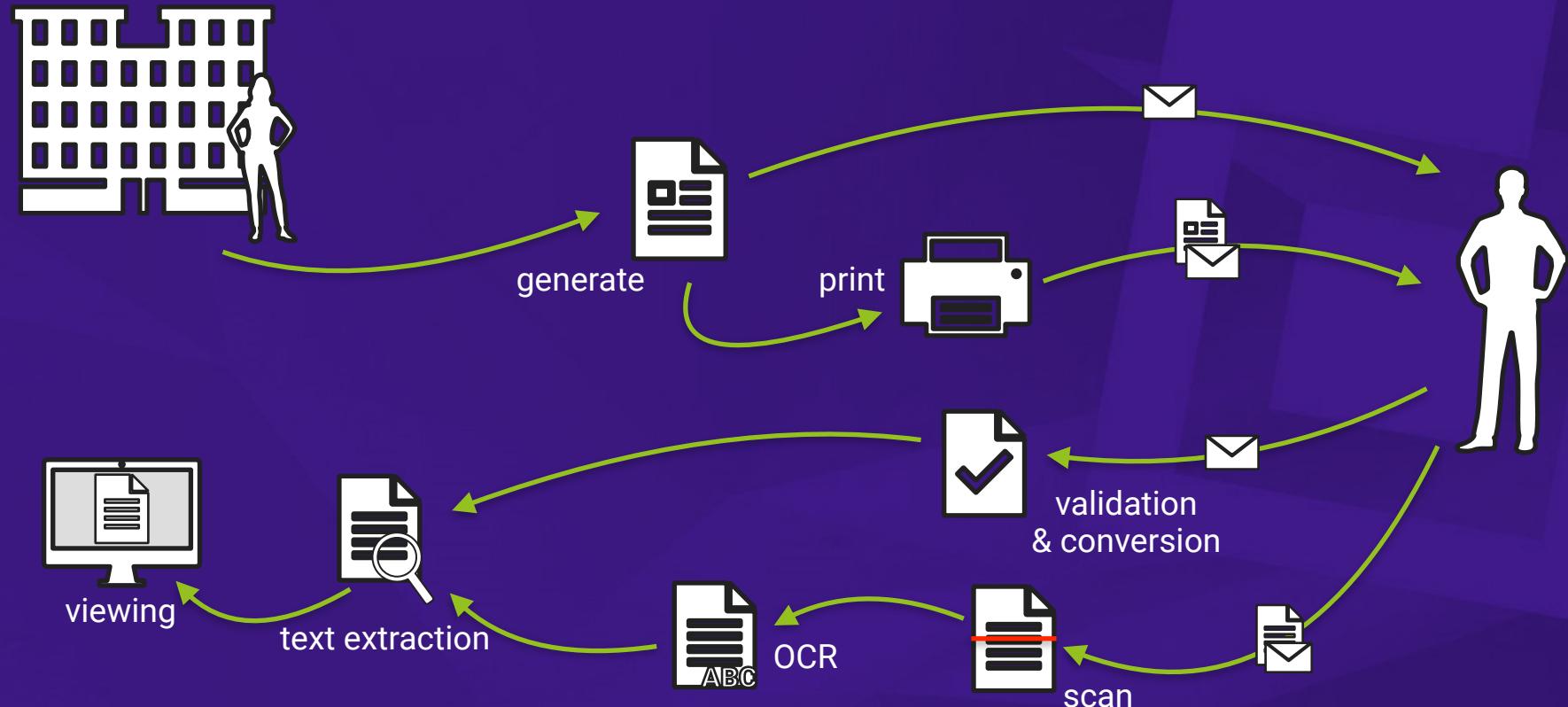
How far can we get?

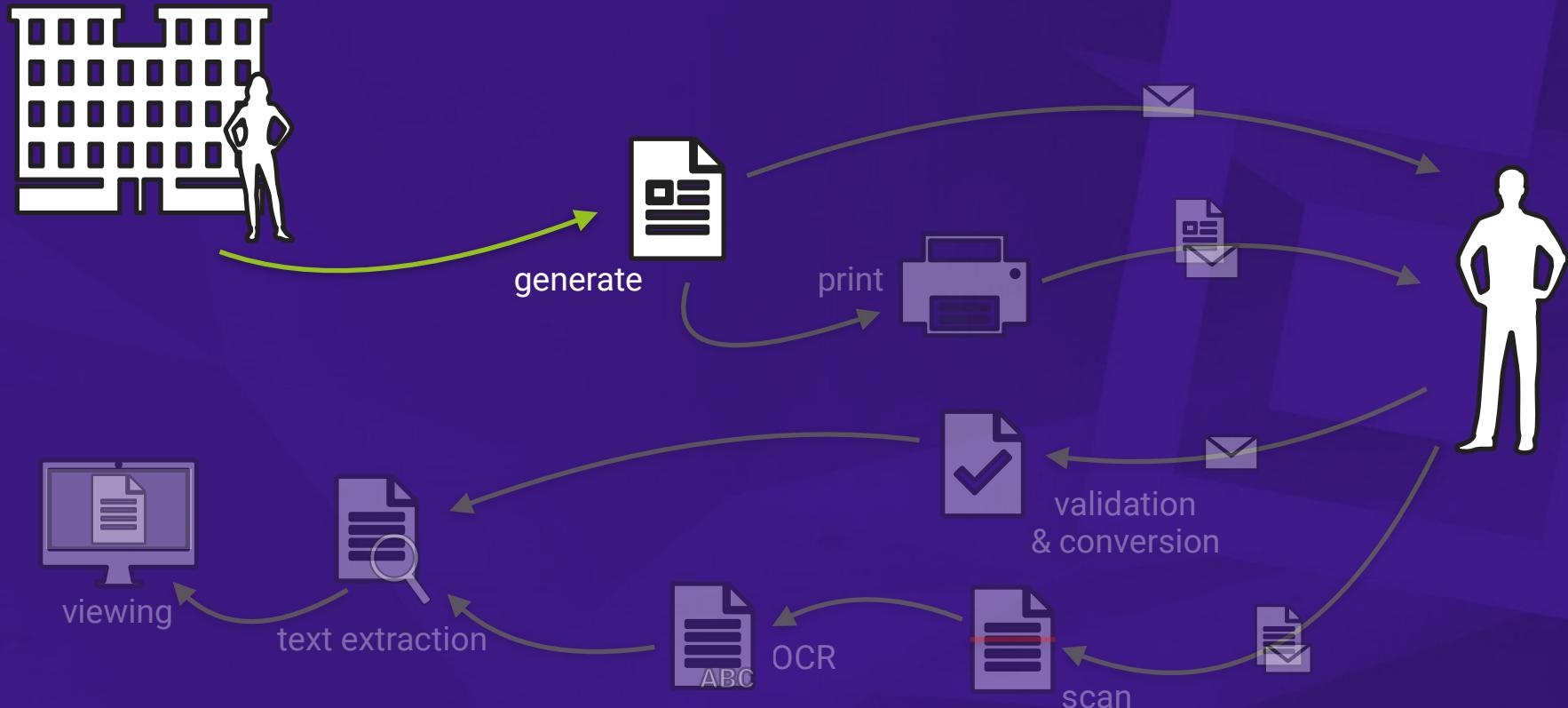
François Fernandès
Senior Solution Architect

francois.fernandes@digitalfrontiers.de
@tellme_francois
GitHub.com/fernansf









GENERATE



Printing



Office
Software



Programmatic
Generation



Declarative
Generation

GENERATE



The screenshot displays four windows of the LibreOffice suite:

- Writer (Word Processing):** Shows a document with placeholder text "Lorem ipsum" and a chart titled "Demo Corp" showing monthly sales data from January to December.
- Calc (Spreadsheets):** Shows a table of the same monthly sales data.
- Impress (Presentations):** Shows a presentation slide with the LibreOffice logo and the number "6".
- Draw (Vector Graphics):** Shows a slide with text and a small diagram.

Word Processing

Spreadsheets



Presentations

GENERATE

- + mature Office Suite
- + capable of generating PDF/A
- + mostly compatible with other Office products

- some compatibility issues when opening documents of other office products

License
Mozilla Public License, Version 2

URL
<https://libreoffice.org/>

Language
C++

Platforms
Windows, MacOS, Linux

GENERATE



Programmatic vs. Declarative

GENERATE

Programmatic vs. Declarative

9

CAPTION

This is a very long sentence that will not fit on a single line.

```
SetFont Arial 50  
MoveTo 100,200  
WriteText CAPTION
```

```
SetFont Arial 20  
MoveTo 100 300  
WriteText This is a very long  
sentence that will not  
fit on a single line.
```

(Extremely-Simplified PDF Instruction Set - ESPIS™)

CAPTION

This is a very long sentence that will not fit on a single line.

```
<Block style="font:Arial;size:50">  
    CAPTION  
</Block>  
  
<Block style="font:Arial;size:20">  
    This is a very long sentence that  
    will not fit on a single line.  
</Block>
```



Apache PDFBox

```
try (PDDocument doc = new PDDocument()) {
    PDPage page = new PDPage();
    doc.addPage(page);

    PDFont font = PDTyp1Font.HELVETICA_BOLD;

    try (PDPageContentStream contents = new PDPageContentStream(doc, page)) {
        contents.beginText();
        contents.setFont(font, 12);
        contents.newLineAtOffset(100, 700);
        contents.showText(message);
        contents.endText();
    }

    doc.save(filename);
}
```



Apache PDFBox

- + mature library
- + capable of generating PDF/A
- + full control of how the document is generated

- steep learning curve

License

Apache License, Version 2

URL

<https://pdfbox.apache.org>

Language

Java

Platforms

Java





XSL:FO

eXtensible Stylesheet Language: Formatting Objects

Programmatic vs. Declarative



GENERATE

XSL:FO

eXtensible Stylesheet Language: Formatting Objects

XML Data

```
<person>
  <name>François Fernandès</name>
  <position>Senior Solution Architect</position>
</person>
```

XSL Stylesheet

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<fo:root>
  <fo:layout-master-set>
    <fo:simple-page-master master-name="A4-portrait"
      page-height="29.7cm" page-width="21.0cm" margin="2cm">
      <fo:region-body/>
    </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="A4-portrait">
    <fo:flow flow-name="xsl-region-body">
      <fo:block>
        Hello, <xsl:value-of select=".//person/name"/>!
      </fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
</xsl:template>
</xsl:stylesheet>
```

FO Document

```
<?xml version="1.0" encoding="utf-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="A4-portrait"
      page-height="29.7cm" page-width="21.0cm" margin="2cm">
      <fo:region-body/>
    </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="A4-portrait">
    <fo:flow flow-name="xsl-region-body">
      <fo:block>
        Hello, François Fernandès!
      </fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```

Hello, François Fernandès!

XSLT Processor





- + based on the XSL:FO Standard
- + capable of generating PDF/A and PDF/X

- moderate community activity
- XSL:FO not fully supported
- XSL:FO has a steep learning curve

License
Apache License, Version 2

URL
<https://xmlgraphics.apache.org/fop/>

Language
Java

Platforms
Java



```
//Initialize PDF writer
PdfWriter writer = new PdfWriter(dest);

//Initialize PDF document
PdfDocument pdf = new PdfDocument(writer);

// Initialize document
Document document = new Document(pdf);

//Add paragraph to the document
document.add(new Paragraph(message));

//Close document
document.close();
```



- + extremely mature library
- + supporting way more than only declarative generation
- + commercial support available
- + available for Java as well as the .net ecosystem

- restrictive license

License

GNU Affero General Public License or
Commercial License

URL

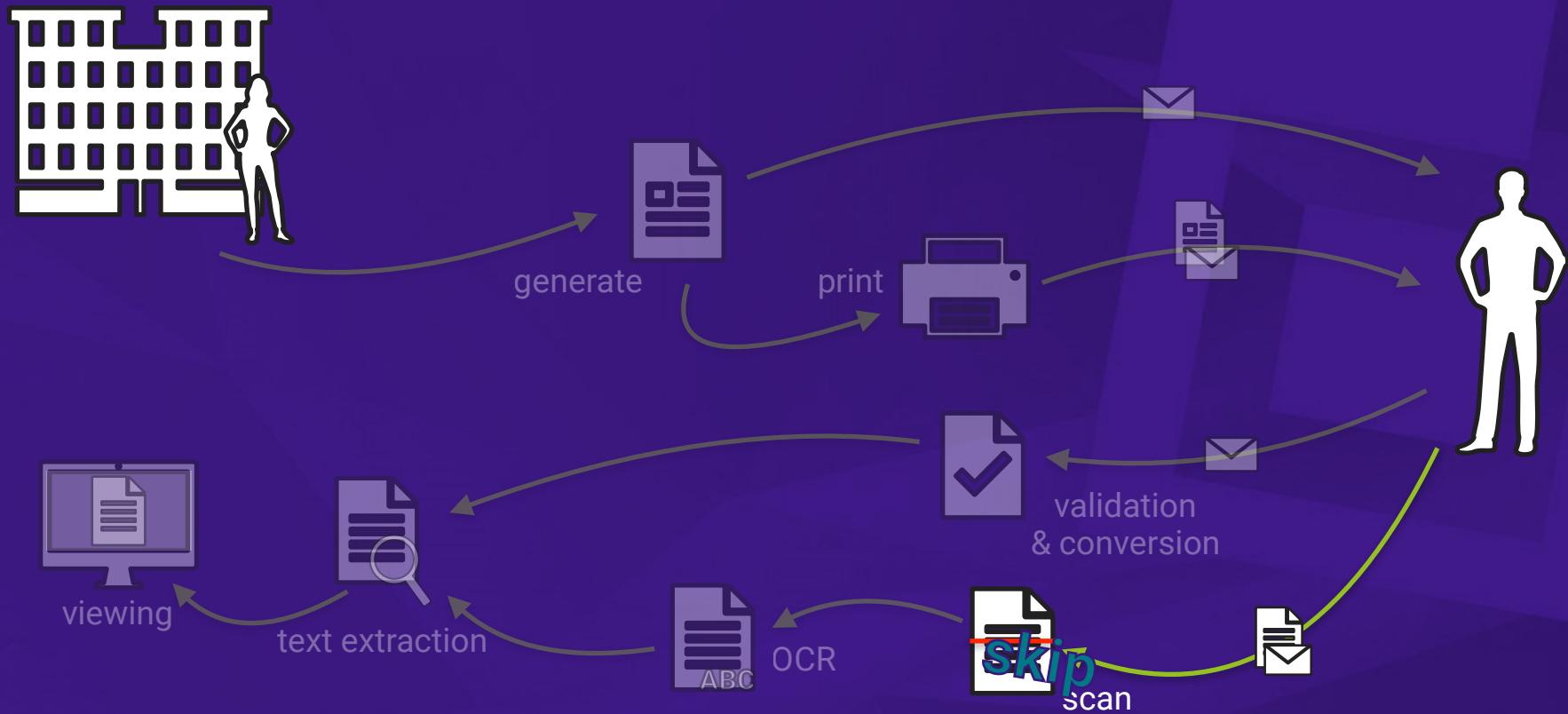
<https://itextpdf.com>

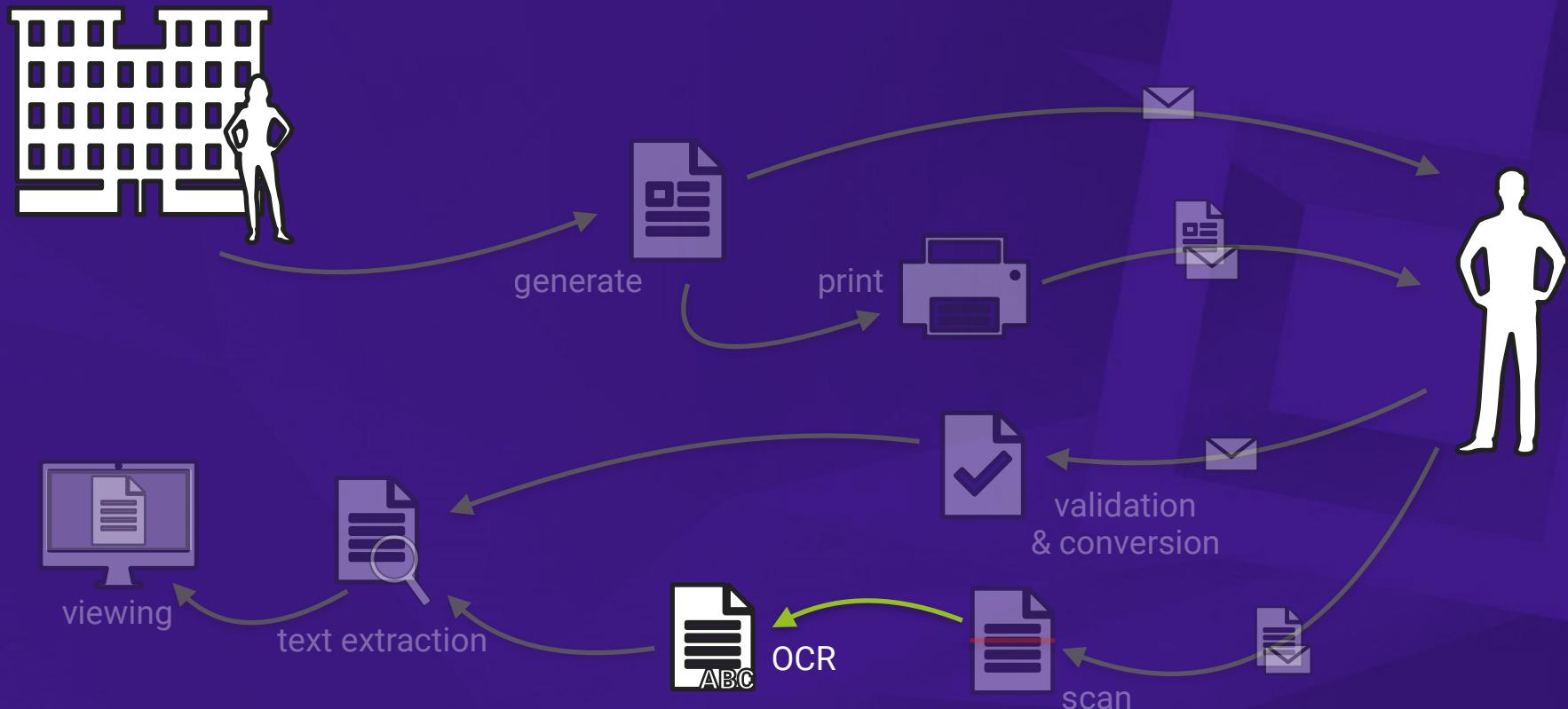
Language

Java, C#

Platforms

Java, .net





OCR

Tesseract OCR



Scan.tiff

tesseract Scan.tiff out -l deu+eng

MITGLIEDSVEREINBARUNG
Physiologisch (Physi) Aktiv
Hiermit erkläre ich meine Mitgliedschaft in der Physiotherapie Vohrer und erhalte damit die Möglichkeit, die Trainingseinrichtung zu nutzen.

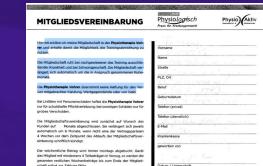
out.txt

tesseract Scan.tiff out -l deu+eng hocr

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<body>
<div class="ocr_page" id="page_1" title="image "Sample.Scan.tif">
<div class="ocr_carea" id="block_1" title="bbox 37 62 394 82">
<p class="ocr_par" id="par_1_1" lang="deu" title="bbox 37 62 394 82">
<span class="ocr_line" id="line_1_1" title="bbox 37 62 394 82">
<span class="ocr_word" id="word_1_1" title="bbox 37 62 394 82">
</span>
</span>
</p>
</div>
</div>
```

out.hocr

tesseract Scan.tiff out -l deu+eng pdf



out.pdf

OCR

Tesseract OCR

- + mature project
- + active community
- + used (and extended) by Google
- + generates searchable PDF

- only accepts images as input

License

Apache License, Version 2.0

URL

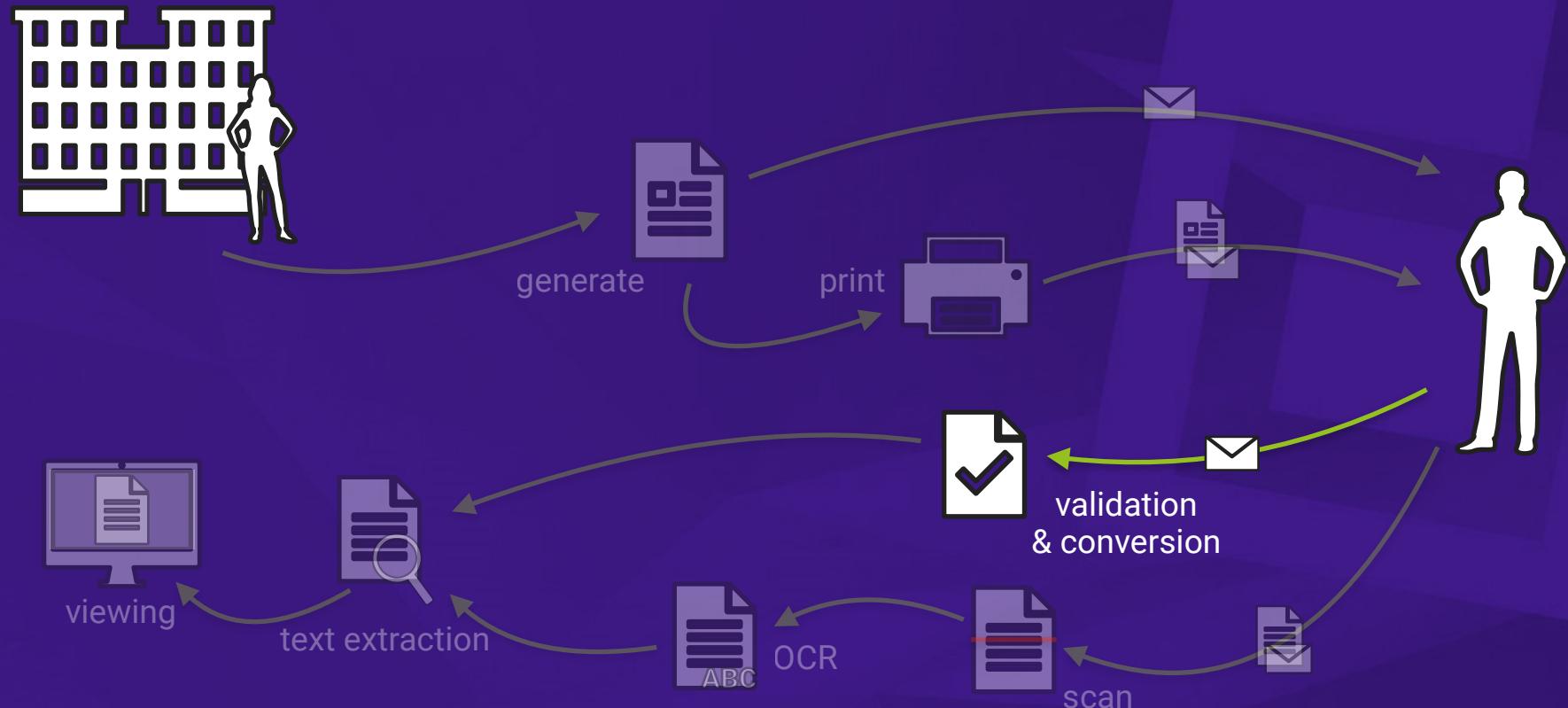
<https://github.com/tesseract-ocr/tesseract>

Language

C++

Platforms

MacOS, Windows, Linux



VALIDATION & CONVERSION



- + active community
- + supported by the PDF Association
- + very extensive coverage of PDF/A-1, PDF/A-2 and PDF/A-3
- only validates PDF/A, not PDF in general

License

GNU General Public License, Version 2 or Mozilla Public License, Version 2

URL

<https://verapdf.org>

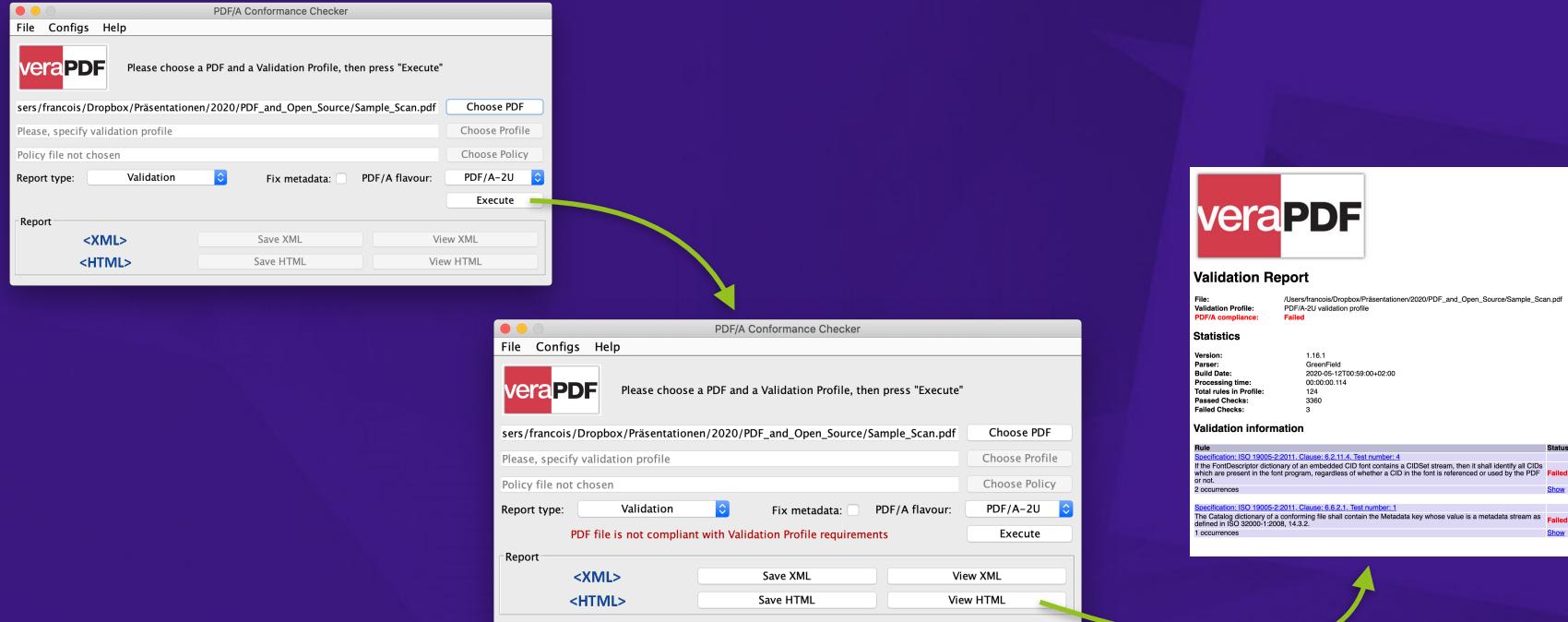
Language

Java

Platforms

MacOS, Windows, Linux

VALIDATION & CONVERSION - veraPDF



VALIDATION & CONVERSION - veraPDF

~/PDF_and_Open_Source # verapdf -f 2b Document.pdf

```

francois@Francoiss-MBP: ~/Dropbox/Präsentationen/2020/PDF_and_Open_Source
> PDF_and_Open_Source verapdf -f 2b Document.pdf
<?xml version="1.0" encoding="utf-8"?>
<report>
  <buildInformation>
    <releaseDetails id="core" version="1.16.1" buildDate="2020-05-12T00:43:00+02:00"></releaseDetails>
    <releaseDetails id="validation-model" version="1.16.1" buildDate="2020-05-12T00:46:00+02:00"></releaseDetails>
    <releaseDetails id="gui" version="1.16.1" buildDate="2020-05-12T00:59:00+02:00"></releaseDetails>
  </buildInformation>
  <job>
    <item size="113546">
      <name>/Users/francois/Dropbox/Präsentationen/2020/PDF_and_Open_Source/Document.pdf</name>
    </item>
    <validationReport profileName="PDF/A-2B validation profile" statement="PDF file is not compliant with Validation Profile requirements." isCompliant="false">
      <details passedRules="120" failedRules="2" passedChecks="3096" failedChecks="3">
        <rule specification="ISO 19005-2:2011" clause="6.2.11.4" testNumber="0" status="failed" passedChecks="0" failedChecks="2">
          <description>If the FontDescriptor dictionary of an embedded CID font contains a CIDSet stream, then it shall identify all CIDs which are present in the font program, regardless of whether a CID in the font is referenced or used by the PDF or not.</description>
          <object>PDCIDFont</object>
          <test>fontfile_size == 0 || fontName.search(/[A-Z]{6}+/) != 0 || CIDSet_size == 0 || cidSetListsAllGlyphs == true</test>
          <check status="failed">
            <context>root/document[0]/pages[0](5 0 obj PDPAGE)/contentStream[0]/operators[792]/font[0](YNEWXU+ArialUnicodeMS)/DescendantFonts[0](GXJCBG+ArialUnicod&lt;...</context>
          </check>
          <check status="failed">
            <context>root/document[0]/pages[0](5 0 obj PDPAGE)/contentStream[0]/operators[32]/font[0](GXJCBG+ArialUnicodeMS)/DescendantFonts[0](GXJCBG+ArialUnicod&lt;...</context>
          </check>
        </rule>
        <rule specification="ISO 19005-2:2011" clause="6.6.2.1" testNumber="1" status="failed" passedChecks="0" failedChecks="1">
          <description>The Catalog dictionary of a conforming file shall contain the Metadata key whose value is a metadata stream as defined in ISO 32000-1:2008, 14.3.2.</description>
          <object>PDdocument</object>
          <test>metadata_size == 1</test>
          <check status="failed">
            <context>root/document[0]</context>
          </check>
        </rule>
      </details>
    </validationReport>
    <duration start="1602753368474" finish="1602753368833">00:00:00.359</duration>
  </job>
  <batchSummary totalJobs="1" failedToParse="0" encrypted="0">
    <validationReports compliant="0" nonCompliant="1" failedJobs="0">1</validationReports>
    <featureReports failedJobs="0">0</featureReports>
    <repairReports failedJobs="0">0</repairReports>
    <duration start="1602753368437" finish="1602753368848">00:00:00.411</duration>
  </batchSummary>
</report>

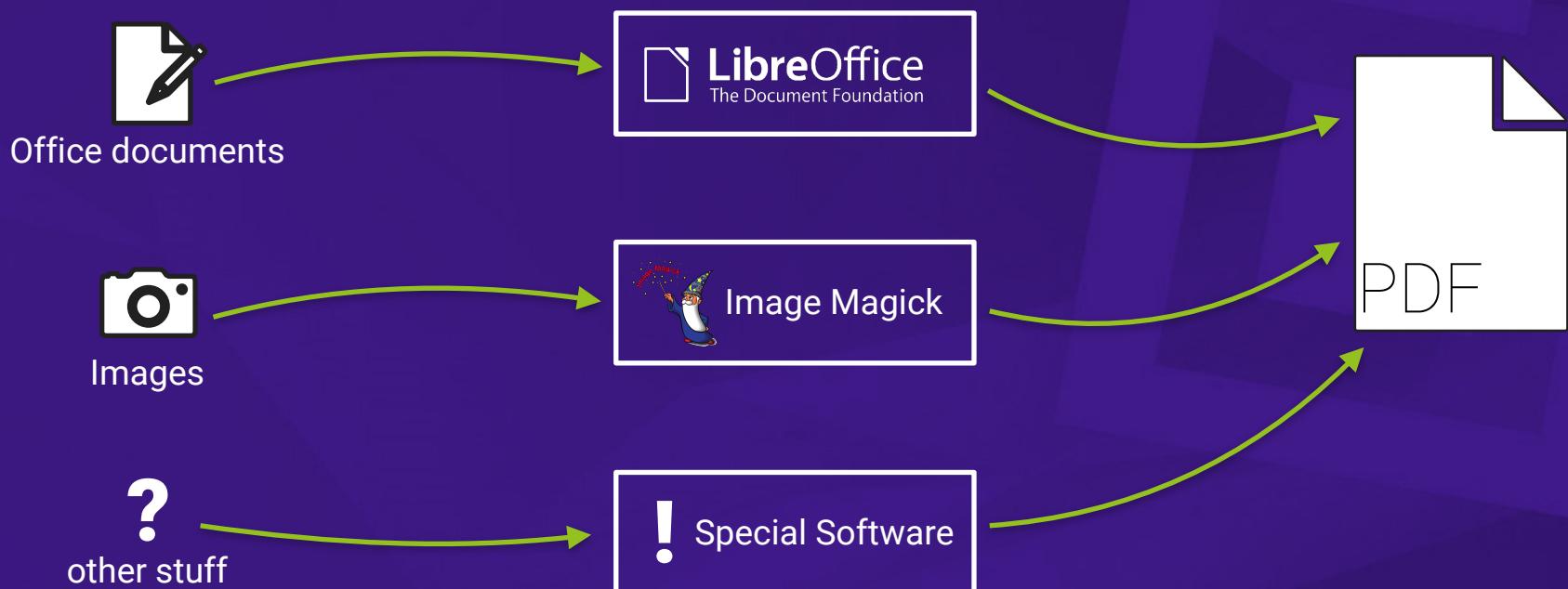
```

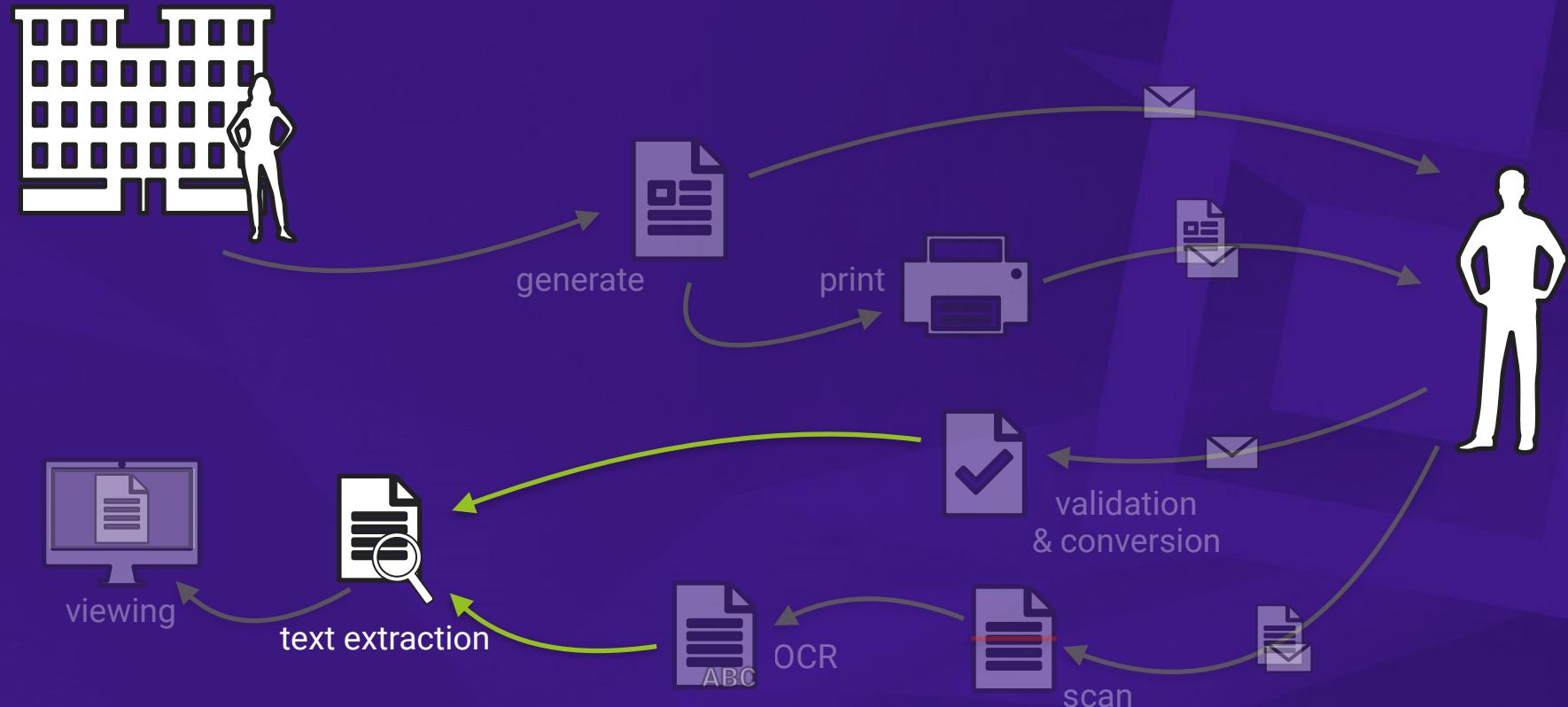
What is the best open source conversion solution?

it depends

none

it depends

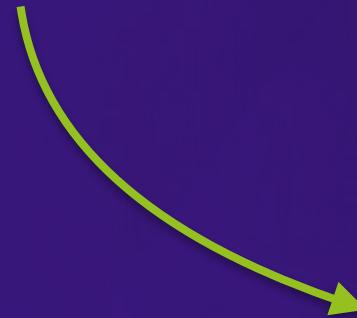




He certainly knows way more than me



Great Library!



October, 20, 2020 16:00

Evaluating Text Extraction at Scale

Case Study from Apache Tika



Presenter: Tim Allison
Language: English

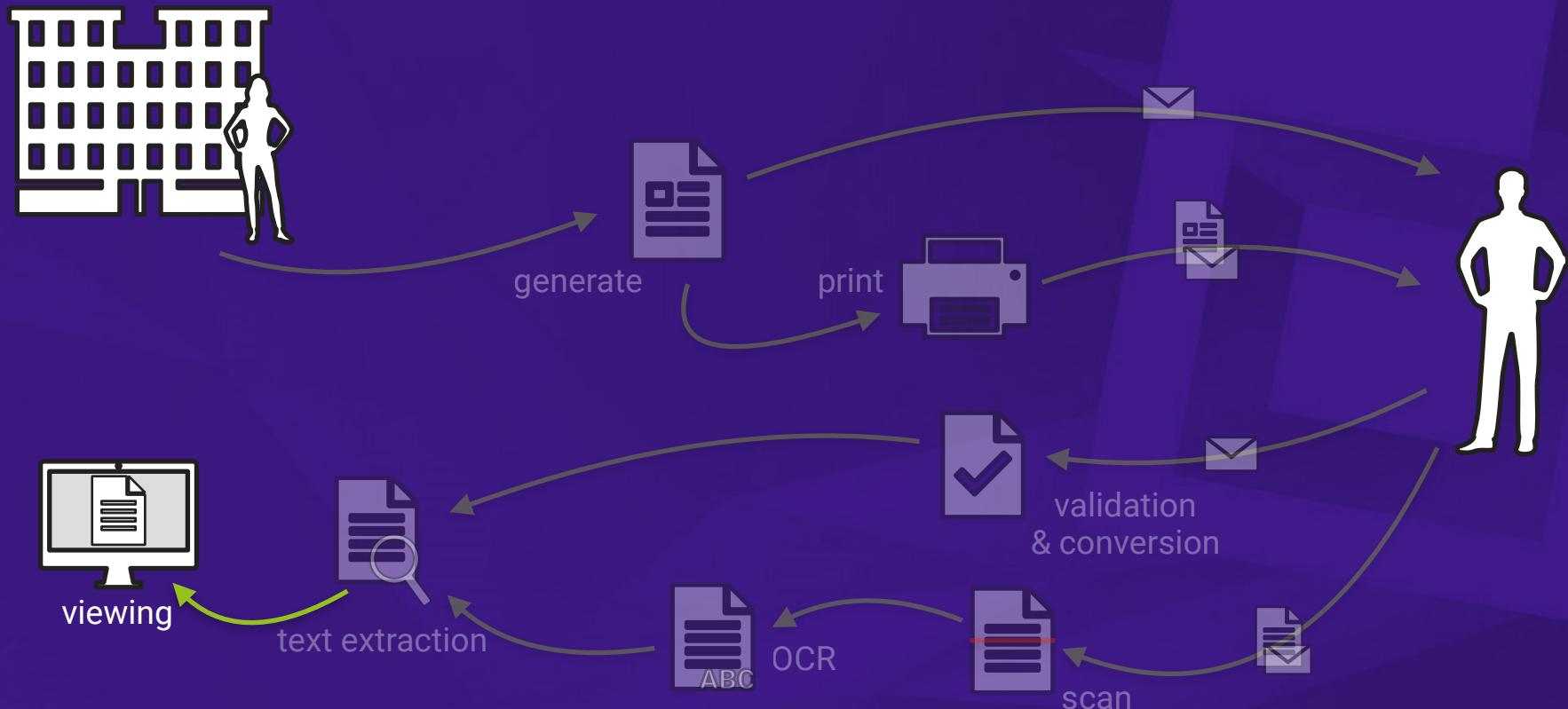
Accessibility Archiving

Description

Apache Tika is widely used as a critical enabling technology for search in Apache Solr and other search systems. This open source library performs text and metadata extraction from numerous file formats, including PDF via an integration with Apache PDFBox. As we all know, when something goes wrong with text extraction, the reliability of search and other natural language processing (NLP) applications is greatly hindered.

Over the last 5 years, the Tika project has gathered and published a large corpus of files (<https://corpora.tika.apache.org/base/docs>), and we have developed an evaluation module (tika-eval) and methodology to identify regressions in text extraction and areas for improvement in our parsers.

This talk offers an overview of Tika's publicly available regression corpus as well as the tika-eval module. We'll discuss ways of scaling its NLP/language-modeling based metrics to identify potential mojibake, corrupt text and/or bad OCR at scale. These techniques have applicability for PDF parser developers, search system integrators and for those interested in archiving and accessibility.



pdf.js

compressed.tracemonkey-pid1

mozilla.github.io/pdf.js/web/viewer.html

1 von 14

- + 70%

Trace-based Just-in-Time Type Specialization for Dynamic Languages

Abstract: PDF.js, Mozilla's open source JavaScript library for displaying PDF files, is a dynamically typed language. It is designed to be fast and efficient, yet it has to support a wide variety of code. One of the main challenges is to generate efficient machine code. It is a statically typed programming language, so type information may vary at runtime. This means that some compiler can no longer rely on static type information to generate efficient code. We used type annotations to help the compiler. We also developed a tracing mechanism to collect type information at runtime. This allows us to generate efficient code for loops. Our tracing mechanism is able to track the execution of loops and determine the type of variables at each iteration. This allows us to generate efficient code for loops. Our tracing mechanism is able to track the execution of loops and determine the type of variables at each iteration. This allows us to generate efficient code for loops.

Categories and Subject Descriptors: D.3.3 [Programming Languages]: Processors—Incremental compilers; code generation; General Terms: Design, Experimentation, Mathematics, Performance.

1. Introduction

Dynamic languages such as JavaScript, Python, and Ruby, are popular because they are expressive, according to type-safety, and make programming easier. However, they are often slower than statically typed languages. One reason for this is that dynamic languages are more difficult to optimize. In this paper, we propose a tracing mechanism to generate efficient machine code for loops. Our tracing mechanism is able to track the execution of loops and determine the type of variables at each iteration. This allows us to generate efficient code for loops.

Permissions and related rights: © 2013, the author(s). This is an open-access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (CC BY-NC), which permits unrestricted reuse and distribution, provided the original author and source are credited.

PLDI'13, June 13–18, 2013, Dublin, Ireland
Copyright © 2013, the author(s). All rights reserved. Licensee ACM, Inc.

Figure 1: Sample program: sieve of Eratosthenes, prime is initialized to an array of 100 false values entry to the code snippet.

1 for (var i = 2; i < 100; ++i) {
2 if (!prime[i])
3 for (var j = i * i; j < 100; k += i)
4 prime[j] = true;
5 }
6
7 prime[0] = prime[1] = false;

Figure 1: Sample program: sieve of Eratosthenes, prime is initialized to an array of 100 false values entry to the code snippet.

1 for (var i = 2; i < 100; ++i) {
2 if (!prime[i])
3 for (var j = i * i; j < 100; k += i)
4 prime[j] = true;
5 }
6
7 prime[0] = prime[1] = false;

compressed.tracemonkey-pid1

mozilla.github.io/pdf.js/web/viewer.html?page=1&zoom=40-62,792

1 von 14

- + 40 %

Trace-based Just-in-Time Type Specialization for Dynamic Languages

Abstract: PDF.js, Mozilla's open source JavaScript library for displaying PDF files, is a dynamically typed language. It is designed to be fast and efficient, yet it has to support a wide variety of code. One of the main challenges is to generate efficient machine code. It is a statically typed programming language, so type information may vary at runtime. This means that some compiler can no longer rely on static type information to generate efficient code. We used type annotations to help the compiler. We also developed a tracing mechanism to collect type information at runtime. This allows us to generate efficient code for loops. Our tracing mechanism is able to track the execution of loops and determine the type of variables at each iteration. This allows us to generate efficient code for loops.

Categories and Subject Descriptors: D.3.3 [Programming Languages]: Processors—Incremental compilers; code generation; General Terms: Design, Experimentation, Mathematics, Performance.

1. Introduction

Dynamic languages such as JavaScript, Python, and Ruby, are popular because they are expressive, according to type-safety, and make programming easier. However, they are often slower than statically typed languages. One reason for this is that dynamic languages are more difficult to optimize. In this paper, we propose a tracing mechanism to generate efficient machine code for loops. Our tracing mechanism is able to track the execution of loops and determine the type of variables at each iteration. This allows us to generate efficient code for loops.

Permissions and related rights: © 2013, the author(s). This is an open-access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (CC BY-NC), which permits unrestricted reuse and distribution, provided the original author and source are credited.

PLDI'13, June 13–18, 2013, Dublin, Ireland
Copyright © 2013, the author(s). All rights reserved. Licensee ACM, Inc.

Figure 1: Sample program: sieve of Eratosthenes, prime is initialized to an array of 100 false values entry to the code snippet.

1 for (var i = 2; i < 100; ++i) {
2 if (!prime[i])
3 for (var j = i * i; j < 100; k += i)
4 prime[j] = true;
5 }
6
7 prime[0] = prime[1] = false;

Figure 1: Sample program: sieve of Eratosthenes, prime is initialized to an array of 100 false values entry to the code snippet.

1 for (var i = 2; i < 100; ++i) {
2 if (!prime[i])
3 for (var j = i * i; j < 100; k += i)
4 prime[j] = true;
5 }
6
7 prime[0] = prime[1] = false;

compressed.tracemonkey-pid1

mozilla.github.io/pdf.js/web/viewer.html?page=1&zoom=40-62,792

1 von 14

- + 40 %

Trace-based Just-in-Time Type Specialization for Dynamic Languages

Abstract: PDF.js, Mozilla's open source JavaScript library for displaying PDF files, is a dynamically typed language. It is designed to be fast and efficient, yet it has to support a wide variety of code. One of the main challenges is to generate efficient machine code. It is a statically typed programming language, so type information may vary at runtime. This means that some compiler can no longer rely on static type information to generate efficient code. We used type annotations to help the compiler. We also developed a tracing mechanism to collect type information at runtime. This allows us to generate efficient code for loops. Our tracing mechanism is able to track the execution of loops and determine the type of variables at each iteration. This allows us to generate efficient code for loops.

Categories and Subject Descriptors: D.3.3 [Programming Languages]: Processors—Incremental compilers; code generation; General Terms: Design, Experimentation, Mathematics, Performance.

1. Introduction

Dynamic languages such as JavaScript, Python, and Ruby, are popular because they are expressive, according to type-safety, and make programming easier. However, they are often slower than statically typed languages. One reason for this is that dynamic languages are more difficult to optimize. In this paper, we propose a tracing mechanism to generate efficient machine code for loops. Our tracing mechanism is able to track the execution of loops and determine the type of variables at each iteration. This allows us to generate efficient code for loops.

Permissions and related rights: © 2013, the author(s). This is an open-access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (CC BY-NC), which permits unrestricted reuse and distribution, provided the original author and source are credited.

PLDI'13, June 13–18, 2013, Dublin, Ireland
Copyright © 2013, the author(s). All rights reserved. Licensee ACM, Inc.

Figure 1: Sample program: sieve of Eratosthenes, prime is initialized to an array of 100 false values entry to the code snippet.

1 for (var i = 2; i < 100; ++i) {
2 if (!prime[i])
3 for (var j = i * i; j < 100; k += i)
4 prime[j] = true;
5 }
6
7 prime[0] = prime[1] = false;

Figure 1: Sample program: sieve of Eratosthenes, prime is initialized to an array of 100 false values entry to the code snippet.

1 for (var i = 2; i < 100; ++i) {
2 if (!prime[i])
3 for (var j = i * i; j < 100; k += i)
4 prime[j] = true;
5 }
6
7 prime[0] = prime[1] = false;

compressed.tracemonkey-pid1

mozilla.github.io/pdf.js/web/viewer.html?page=1&zoom=40-62,792

1 von 14

- + 40 %

Trace-based Just-in-Time Type Specialization for Dynamic Languages

Abstract: PDF.js, Mozilla's open source JavaScript library for displaying PDF files, is a dynamically typed language. It is designed to be fast and efficient, yet it has to support a wide variety of code. One of the main challenges is to generate efficient machine code. It is a statically typed programming language, so type information may vary at runtime. This means that some compiler can no longer rely on static type information to generate efficient code. We used type annotations to help the compiler. We also developed a tracing mechanism to collect type information at runtime. This allows us to generate efficient code for loops. Our tracing mechanism is able to track the execution of loops and determine the type of variables at each iteration. This allows us to generate efficient code for loops.

Categories and Subject Descriptors: D.3.3 [Programming Languages]: Processors—Incremental compilers; code generation; General Terms: Design, Experimentation, Mathematics, Performance.

1. Introduction

Dynamic languages such as JavaScript, Python, and Ruby, are popular because they are expressive, according to type-safety, and make programming easier. However, they are often slower than statically typed languages. One reason for this is that dynamic languages are more difficult to optimize. In this paper, we propose a tracing mechanism to generate efficient machine code for loops. Our tracing mechanism is able to track the execution of loops and determine the type of variables at each iteration. This allows us to generate efficient code for loops.

Permissions and related rights: © 2013, the author(s). This is an open-access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (CC BY-NC), which permits unrestricted reuse and distribution, provided the original author and source are credited.

PLDI'13, June 13–18, 2013, Dublin, Ireland
Copyright © 2013, the author(s). All rights reserved. Licensee ACM, Inc.

Figure 1: Sample program: sieve of Eratosthenes, prime is initialized to an array of 100 false values entry to the code snippet.

1 for (var i = 2; i < 100; ++i) {
2 if (!prime[i])
3 for (var j = i * i; j < 100; k += i)
4 prime[j] = true;
5 }
6
7 prime[0] = prime[1] = false;

Figure 1: Sample program: sieve of Eratosthenes, prime is initialized to an array of 100 false values entry to the code snippet.

1 for (var i = 2; i < 100; ++i) {
2 if (!prime[i])
3 for (var j = i * i; j < 100; k += i)
4 prime[j] = true;
5 }
6
7 prime[0] = prime[1] = false;

VIEWING

pdf.js

- + active community
- + used in Firefox

- struggles with some advanced PDF features
- limited by Java Script

License

Apache License, Version 2

URL

<https://mozilla.github.io/pdf.js/>

Language

Java Script

Platforms

any modern Browser

VIEWING



VIEWING



based on



VIEWING



based on



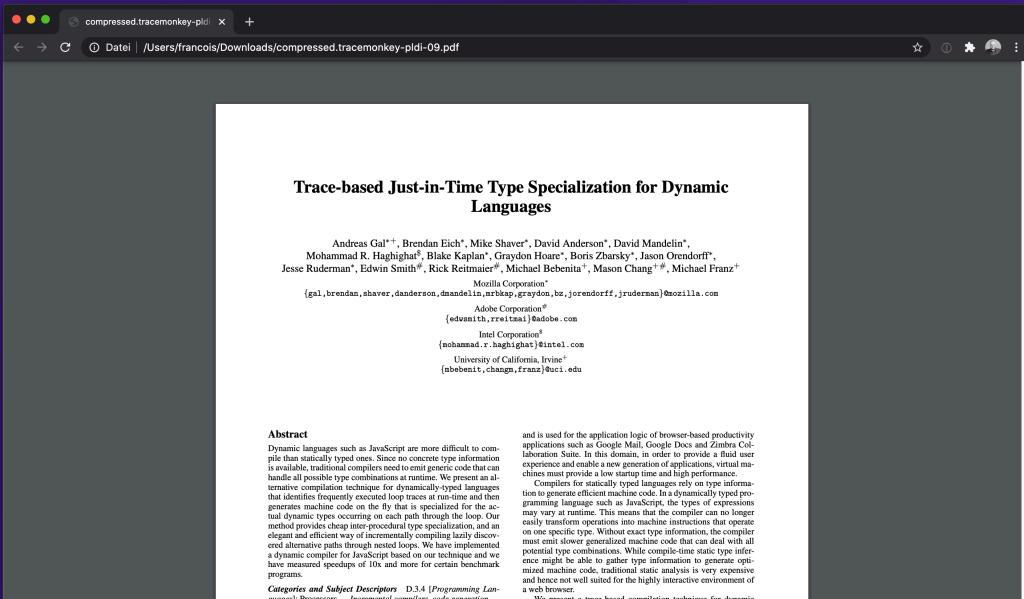
includes

PDFium

The PDFium logo, which is a white rounded rectangle with a green dashed border containing the word "PDFium".

VIEWING

PDFium



Abstract

Dynamic languages such as JavaScript are more difficult to compile than statically typed ones. Since no concrete type information is available, traditional compilers must emit generic code that can handle all possible types. In contrast, we propose a trace-based alternative compilation technique for dynamically-typed languages that identifies frequently executed loop traces at run-time and then generates specialized code for them. Our approach handles actual dynamic types occurring on each path through the loop. Our method provides cheap inter-procedural type specialization, and an elegant and efficient way of incrementally specializing heavily-discretized control paths as they are nested by loops. We have implemented a dynamic compiler for JavaScript based on our technique and we have measured speedups of 10x and more for certain benchmarks.

Categories and Subject Descriptors D.3.4 (Programming Languages): Processors — Incremental compilers; code generation

and is used for the application logic of browser-based productivity applications such as Google Mail, Google Docs and Zimbra Collaboration Suite. In this domain, in order to provide a fluid user experience, the application logic must be fast. Traditional compilers must provide a low startup time and high performance.

Compilers for statically typed languages typically rely on type information generated at compile time. In addition, most programming languages such as JavaScript, the types of expressions may vary at runtime. This means that the compiler can no longer easily translate operations into machine instructions that operate on one specific type. Without exact type information, the compiler must emit slower generalized machine code that can deal with all potential type combinations. While compile-time static type inference is a well-known technique for statically typed languages, in optimized machine code, traditional static analysis is very expensive and hence not well suited for the highly interactive environment of a web browser.

We present a trace-based compilation technique for dynamic

VIEWING

PDFium

- + active community
- + widely available
- + PDFium is available as a library

- not many customization options in Chrom{e,ium}

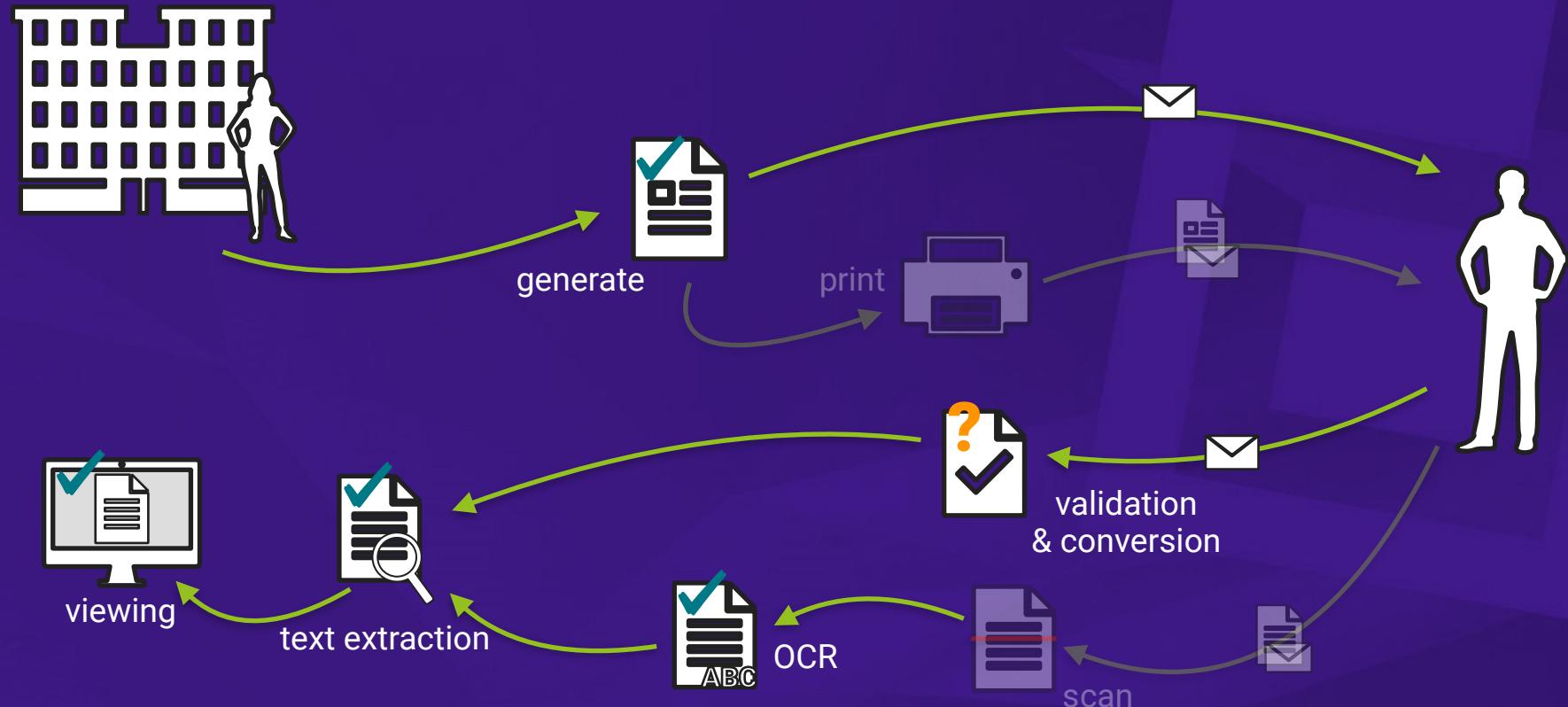
License
BSD-Style

URL
<http://pdfium.org>

Language
C++

Platforms
MacOS, Windows, Linux

Recap





What is your next PDF project?

40

François Fernandès

Senior Solution Architect



@tellme_francois



github.com/fernansfs

<https://blog.digitalfrontiers.de>

<https://www.digitalfrontiers.de>

#WeAreHiring



digital
frontiers