SERVICE MEISTER

INHALTSEXTRAKTION UND LOGISCHE STRUKTUR-ERKENNUNG IN PDF-DOKUMENTEN

Whitepaper

Extraktion von Texten, Tabellen und Bildern aus Handbüchern im Service-Meister Forschungsprojekt.

Ein Beitrag der Konsortialpartner KROHNE und inovex.

KROHNE measure the facts

inovex

ANSPRECHPARTNER:

- Dr. Robert Pesch (inovex, rpesch@inovex.de)
- Dr. Martin Krawczyk-Becker (KROHNE Messtechnik, M.Krawczyk-Becker@KROHNE.com)

Inhalt

ANSPRECHPARTNER:
- Dr. Robert Pesch (inovex, rpesch@inovex.de)
- Dr. Martin Krawczyk-Becker (KROHNE Messtechnik, M.Krawczyk-Becker@KROHNE.com)

Management Summary

Egal, ob Handbücher, Bedienungs- und Reparaturanleitungen oder Wartungsberichte – liegen sie im Portable Document Format (PDF) vor, lassen sich nur auf Umwegen Texte, Bilder, Tabellen oder Inhaltsverzeichnisse daraus extrahieren. Warum das so ist: Weil PDFs rein für die Darstellung entwickelt sind.

Wer die Dokumente maschinell auswerten möchte, um Informationen automatisiert weiterzuverarbeiten, der greift auf Python-Bibliotheken zurück. Viele Bibliotheken bieten sich an, um PDF-Inhalte zu extrahieren und strukturiert abzulegen. Aber: Die Extraktion von Text, Bildern und Tabellen ist aufgrund der Schwierigkeiten, die das PDF-Format mit sich bringt, alles andere als einfach und funktioniert auch mit geeigneten Bibliotheken nicht komplett fehlerfrei.

Das Whitepaper zeigt auf, welche Ansätze sich anbieten, um relevante Inhalte aus den PDF-Dokumenten zu ziehen und deren logische Struktur zu erkennen. Dazu vergleicht und bewertet die Veröffentlichung die Leistungsfähigkeit unterschiedlicher Python-Bibliotheken vor dem Hintergrund der Anforderungen, die sich aus dem KI-Projekt Service-Meister ergeben. Das Ziel: Inhalte sollen sich weitgehend automatisch extrahieren und maschinenlesbar etwa in XML- und JSON-Formaten bereitstellen lassen.

ANSPRECHPARTNER:
- **Dr. Robert Pesch (inovex, rpesch@inovex.de)**
- **Dr. Martin Krawczyk-Becker (KROHNE Messtechnik, M.Krawczyk-Becker@KROHNE.com)**

Diese Arbeit wurde im Rahmen des Forschungsprojekts „Service-Meister“ durchgeführt und gefördert vom Bundeswirtschaftsministerium (BMWK).

Das Portable Document Format (PDF) ist das meistgenutzte Austauschformat für Dokumente. Im Gegensatz zu Dokumenten im strukturierten Extensible Markup Language Format (XML) oder im JavaScript Object Notation Format (JSON) weisen PDF-Dokumente keine maschinellelesbaren Strukturmarker auf, die beispielsweise definieren, wo ein neuer Abschnitt anfängt oder ob es sich bei einer bestimmten Textfolge um eine Tabelle handelt. PDF-Dokumente enthalten lediglich Anweisungen, wie die Seite geschrieben wird. Sätze, Paragraphen und Kapitel werden als eine Menge von Zeichen repräsentiert, deren Reihenfolge im Dokument nicht mit der Ordnung in der grafischen Ausgabe übereinstimmen muss.

Bei der Extraktion von Inhalten aus Dokumenten kommt es daher oft zu Problemen. Das Erkennen von Wörtern und Paragraphen muss über Heuristiken durchgeführt werden und erfolgt meist basierend auf Abständen zwischen den Zeichen, was jedoch Fehler mit sich bringt. Häufig gibt es Zeichen, die zwar nicht sichtbar sind, aber extrahiert werden, z.B. weil der Text in letzter Minute noch angepasst wurde. Außerdem treten, beispielsweise in Überschriften, zusätzliche Leerzeichen zwischen Zeichen auf.

Diese dienen zur besseren Darstellung aber sorgen für eine verfälschte Extraktion. Andersherum fehlen oft Leerzeichen zwischen Wörtern. Auch Sonderzeichen werden häufig nicht richtig codiert und falsch extrahiert. Hinterlegte Metadaten, beispielsweise über den Autor, das Erstelldatum und die vorhandenen Kapitel, sind oftmals fehlerhaft oder überhaupt nicht vorhanden. Da PDF-Dokumente zur Darstellung konzipiert wurden, werden Tabellen und Bilder nicht gesondert gespeichert. Deswegen ist es schwierig Bilder und Tabellen korrekt zu extrahieren und deren Position im Gesamtkontext zu bestimmen.

Im Service-Meister Forschungsprojekt erstellt inovex gemeinsam mit Krohne, einem führenden Anbieter für Prozessmesstechnik, in einem geschlossenen Teilvorhaben KI-Applikationen für die Wasser- und Abwasserwirtschaft. In diesem Teilvorhaben sollen u.a. Servicetechniker bei Wartungs- und Reparaturarbeiten unterstützt werden. Hierfür sollen relevante Informationen aus Serviceberichten, Handbüchen, Anleitungen und allgemeine Geräteinformationen gezielt durchsuchbar gemacht werden.

Dafür ist es notwendig, relevante Textabschnitte, Tabellen und Bilder aus den Dokumenten zu extrahieren, mit externen Informationen anzureichern und effizient durchsuchbar zu machen. Alle relevanten Dokumente liegen in unserem Use-Case ausschließlich als PDF-Dokumente vor und beinhalten die oben aufgelisteten Probleme.

ANSPRECHPARTNER:
- **Dr. Robert Pesch (inovex, rpesch@inovex.de)**
- **Dr. Martin Krawczyk-Becker (KROHNE Messtechnik, M.Krawczyk-Becker@KROHNE.com)**

1 Python-Bibliotheken für die Extraktion von PDF-Inhalten

Für die Extraktion von Inhalten aus PDF-Dokumenten stehen viele frei benutzbare Python-Bibliotheken zur Verfügung. Im Detail haben wir uns die folgenden PDF-Bibliotheken für die Extraktion von Text, Bildern, Inhaltsverzeichnissen und Tabellen angeschaut: PDFMiner, Apache Tika, PyPDF, PyMuPDF, Camelot, tabula-py und PDFPlumber. In Tabelle 1 befindet sich eine Übersicht der Bibliotheken mit ihren für uns relevanten Funktionsumfängen.

	Text extraktion	Bild- extraktion	Inhaltsverzeichniss- extraktion	Tabellen-extraktion	Lizenz
PDFMiner.six https://github.com/pdfminer/pdfminer.six	Ja	Ja	Ja	Nein	MIT
Python wrapper für Apache Tika https://github.com/chrismattmann/tika-python	Ja	Ja	Nein	Nein	Apache-2.0
PyPDF2 und PyPDF4 https://github.com/mstamy2/PyPDF2	Ja	Ja	Ja	Nein	BSD-3-Clause
PyMuPDF https://github.com/pymupdf/PyMuPDF	Ja	Ja	Ja	Nein	GPL-3.0
Camelot https://github.com/camelot-dev/camelot	Nein	Nein	Nein	Ja	MIT
tabula-py https://github.com/chezou/tabula-py	Nein	Nein	Nein	Ja	MIT
PDFPlumber https://github.com/jsvine/pdfplumber	Ja	Ja	Nein	Ja	MIT

Tabelle 1: Übersicht der getesteten Python Bibliotheken zur Extraktion von PDF-Inhalten.

2 Bibliotheken für die Text- und Bildextraktion:

Zur Extraktion von Text und Bildern wurden die Python-Bibliotheken PDFMiner, Apache Tika, PyPDF, PyMuPDF und PDFPlumber miteinander verglichen. Mit allen Bibliotheken konnte der Text erfolgreich extrahiert werden. Mit PDFMiner konnten wir bereits mit Standardparametern in unserem konkreten Anwendungsfall die beste Textextraktion erzielen. Außerdem wurde die Reihenfolge mit Ausnahme von Kopf- und Fußzeile meist in der richtigen Reihenfolge extrahiert. Durch eine Vielzahl von Parametern ist es außerdem möglich, die Extraktion weiter anzupassen und zu verbessern. Außerdem gibt es viele Quellen zur Nutzung von PDFMiner im Internet.

ANSPRECHPARTNER:
- **Dr. Robert Pesch (inovex, rpesch@inovex.de)**
- **Dr. Martin Krawczyk-Becker (KROHNE Messtechnik, M.Krawczyk-Becker@KROHNE.com)**

Bilder können als Rastergrafik oder als Vektorgrafik im PDF-Dokument hinterlegt werden. Vektorgrafiken konnten weder mit den oben genannten Bibliotheken noch mit webbasierten Tools wie beispielsweise PDFCandy, PDFaid, ExtractPDF und pdf-online erkannt und extrahiert werden. Rastergrafiken konnten wir in unseren Dokumenten mit PDFMiner, Apache Tika, PyPDF, PyMuPDF und PDFPlumber erfolgreich extrahieren. Aufgrund der guten Ergebnisse bei der Textextraktion, der ausführlichen Dokumenten und der aktiven Community haben wir uns entschieden PDFMiner für die Bild und Textextraktion zu verwenden.

3 Bibliotheken für die Tabellenextraktion

Für die Tabellenextraktion eignen sich Camelot, tabula-py und PDFPlumber, da es mit ihnen möglich ist Tabellen explizit z.B. als pandas DataFrame zu extrahieren. Die Extraktionsqualität mit Camelot ist dabei bereits mit Standardparametern besser als bei tabula-py und PDFPlumber. Außerdem können viele hilfreiche Parameter angepasst werden. Aus diesen Gründen haben wir uns entschieden Camelot zur Tabellenextraktion zu verwenden.

Besonders nützliche Parameter bei Camelot sind der Accuracy Parameter (welcher die Korrektheit der Extraktion approximiert) und der Whitespace Parameter (welcher angibt, wieviel Prozent der Tabelle leer sind). Mithilfe dieser Parameter können viele inkorrekt extrahierte Tabellen erkannt und entfernt werden. Ein Nachteil von Camelot ist die schlechte Performance beim Extrahieren von Tabellen- und Spaltennamen. Falls keine gesonderte Textextraktion zur Bestimmung der Namen erfolgt, könnte PDFPlumber in solch einem Fall die bessere Wahl sein.

4 Bibliotheken für die Extraktion des Inhaltsverzeichnisses

Da PDFMiner und PyMuPDF bereits für die Text- und Bildextraktion sehr gute Ergebnisse geliefert haben und die Dokumentation bei PyPDF nicht sehr umfassend ist konzentrierte sich der Vergleich bei der Inhaltsverzeichnisextraktion auf PDFMiner und PyMuPDF. Da alle gängigen Python-Bibliotheken das Inhaltsverzeichnis basierend auf den Metadaten extrahieren, funktioniert die Extraktion nur dann, wenn diese Daten korrekt hinterlegt sind. Obwohl die Extraktionsqualität bei beiden Bibliotheken sehr ähnlich war, entschieden wir uns für PyMuPDF, da es hier sehr einfach ist die Seitenzahlen der Kapitel auszulesen und generell weniger Code benötigt wird.

5 Informationsextraktion aus PDF-Dokumenten im Service-Meister-Forschungsprojekt

Die PDF-Dokumente bestehen aus Kapiteln, welche wiederum aus Unterkapiteln bestehen. Diese beinhalten Textelemente, Tabellen und Bilder, welche mithilfe der unterschiedlichen Bibliotheken extrahiert werden müssen. Ein Ausschnitt aus dem PDF-Dokument befindet sich in Abbildung 1.

ANSPRECHPARTNER:
- **Dr. Robert Pesch (inovex, rpesch@inovex.de)**
- **Dr. Martin Krawczyk-Becker (KROHNE Messtechnik, M.Krawczyk-Becker@KROHNE.com)**

Hauptkapitel

Produktname

6 TECHNICAL DATA

OPTIFLUX 1000

Unterkapitel

6.3 Measuring accuracy

Text

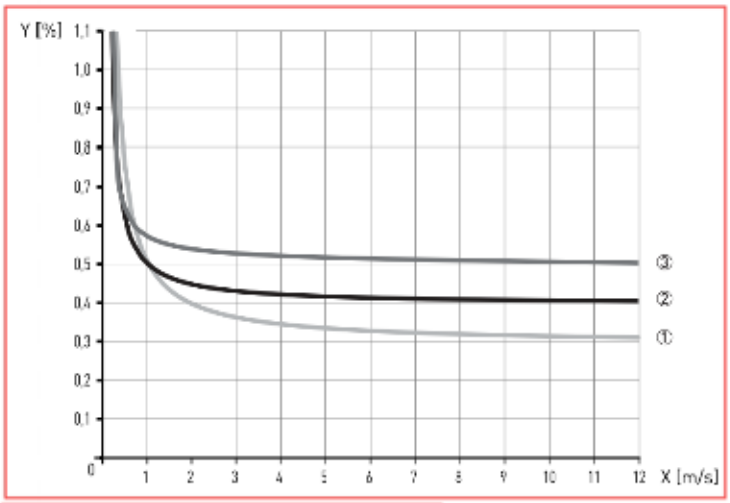
Every electromagnetic flowmeter is calibrated by direct volume comparison. The wet calibration validates the performance of the flowmeter under reference conditions against accuracy limits.

The accuracy limits of electromagnetic flowmeters are typically the result of the combined effect of linearity, zero point stability and calibration uncertainty.

Reference conditions

- Medium: water
- Temperature: +5...35°C / +41...95°F
- Operating pressure: 0.1...5 barg / 1.5...72.5 psig
- Inlet section: ≥ 5 DN
- Outlet section: ≥ 2 DN

Bild



Text

Figure 6-2: Flow velocity vs. accuracy

X [m/s]: flow velocity
Y [%]: deviation from the actual measured value (mv)

Accuracy

Tabelle

Flow sensor diameter	Signal converter type	Accuracy	Curve
DN10...150 [3/8...6"]	IFC 050	0.5% of mv + 1 mm/s	③
DN10...150 [3/8...6"]	IFC 100	0.4% of mv + 1 mm/s	②
DN10...150 [3/8...6"]	IFC 300	0.3% of mv + 2 mm/s	①

Abbild 1: Ausschnitt einer Seite aus dem Krohne-Handbuch im PDF-Format. (Seite 30, Optiflux 1000).

ANSPRECHPARTNER:
- Dr. Robert Pesch (inovex, rpesch@inovex.de)
- Dr. Martin Krawczyk-Becker (KROHNE Messtechnik, M.Krawczyk-Becker@KROHNE.com)

Die extrahierten Elemente überführen wir in ein strukturiertes JSON-Format und speichern sie dann in Elasticsearch, um eine effiziente Suche zu ermöglichen.

6 Extraktion von Texten

Als Ausgabe erhalten wir durch PDFMiner einen zusammenhängenden String einer Seite oder des gesamten PDF-Dokuments. Wir iterieren über jede Seite einzeln und zerlegen den Textblock mithilfe der spaCy Bibliothek in Sätze. Daraufhin trennen wir Sätze mit doppeltem Zeilenumbruch (\n\n) und Sätze mit einfachem Zeilenumbruch, die mit keinem gängigen Satzende wie Punkt oder Ausrufezeichen enden. Außerdem entfernen wir überflüssige Leerzeichen. Schlussendlich entfernen wir alle verbleibenden Zeilenumbrüche in Sätzen, da diese in der finalen Ausgabe nicht mehr benötigt werden.

Beispiel:
Abbildung 2 zeigt einen Ausschnitt aus dem Krohne-Handbuch, dessen Text extrahiert werden soll.

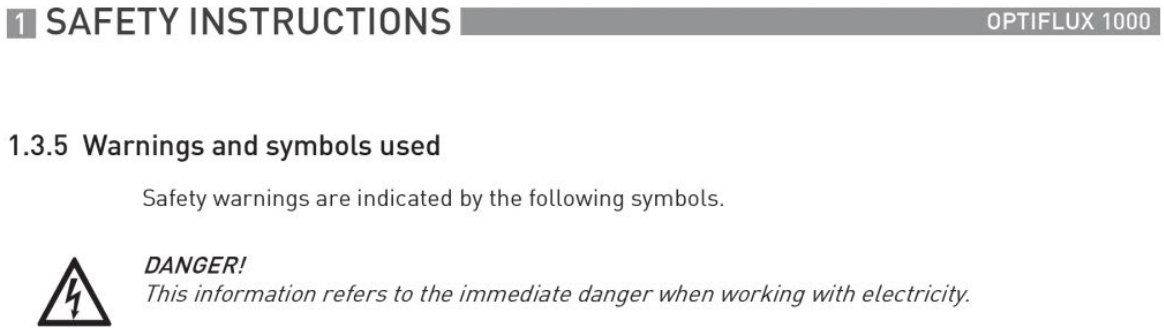


Abbildung 2: Zu extrahierender Text aus dem Krohne-Handbuch (Seite 8, Optiflux 1000).

Nach der Extraktion durch PDFMiner erhalten wir den folgenden zusammenhängenden Textblock:

„1 SAFETY INSTRUCTIONS \n\nOPTIFLUX 1000\n\n1.3.5 Warnings and symbols used\n\nSafety warnings are indicated by the following symbols.\n\nDANGER!\nThis information refers to the immediate danger when working with electricity.”

Nach der zusätzliche Verarbeitung wird der Text in einzelne Einträge aufgeteilt und sieht wie folgt aus:

‚1 SAFETY INSTRUCTIONS‘

‚OPTIFLUX 1000‘

‚1.3.5 Warnings and symbols used‘

‚Safety warnings are indicated by the following symbols.‘

‚DANGER!‘

‚This information refers to the immediate danger when working with electricity.‘

ANSPRECHPARTNER:
- Dr. Robert Pesch (inovex, rpesch@inovex.de)
- Dr. Martin Krawczyk-Becker (KROHNE Messtechnik, M.Krawczyk-Becker@KROHNE.com)

7 Extraktion von Bildern

Um das Problem von kleinen Bildern, die wir nicht extrahieren wollen, zu lösen, filtern wir basierend auf der Bild- und Dateigröße. Die extrahierten Rastergrafiken speichern wir als Base64-String und als Bild-Embeddings (hochdimensionale numerische Bilddarstellungen), um so in Zukunft einen Ähnlichkeitsvergleich zwischen den Bildern zu ermöglichen.

8 Extraktion von Tabellen

Nach der Extraktion durch Camelot filtern wir die Tabellen basierend auf dem Accuracy Parameter (welcher die Korrektheit der Extraktion approximiert) und dem Whitespace Parameter (welcher angibt wieviel Prozent der Tabelle leer sind). Wir behalten nur Tabellen, die eine Accuracy > 75 und maximal 90% Whitespace haben. Außerdem entfernen wir Tabellen, die aus einer einzigen Spalten bestehen, da es sich hierbei um Fehler in der Extraktion handelt. Alle verbleibenden Tabellen werden als CSV-Dateien gespeichert.

9 Extraktion von Inhaltsverzeichnissen

Zur Extraktion von Inhaltsverzeichnissen (IHV) verwenden wir die Python-Bibliothek PyMuPDF, falls die Kapitelinformationen in den Metadaten gespeichert sind. Dies ist jedoch leider nur bei wenigen PDF-Dokumenten der Fall. Außerdem sind die gespeicherten Kapitelinformationen oft falsch und beziehen sich auf einzelne Textabschnitte mit falschen Seitenzahlen statt auf Kapitel. PyMuPDF verwenden wir deswegen nur dann, wenn Kapitelinformationen existieren und jeder Eintrag eine sinnvolle dazugehörige Seitenzahl (ungleich -1) beinhaltet. Ist dies nicht der Fall, verwenden wir einen eigenen Ansatz, der auf dem bereits extrahierten Text basiert:

Dafür müssen wir als Erstes herausfinden, welche Seiten zum IHV gehören. Falls bereits ein Teil des IHV identifiziert wurde, ignorieren wir Seiten, deren vorherige Seite nicht bereits Teil des IHV war, da das IHV ein zusammenhänges Kapitel ist. Wir bezeichnen eine Seite nur als Teil des IHV, wenn mindestens 30% der Zeilen typische Kriterien eines Inhaltsverzeichniseintrags erfüllen. Dazu gehört beispielsweise, dass der Eintrag am Ende eine Zahl hat, welche die Startseite des Kapitels angibt. Außerdem muss sich die Startseite innerhalb des Dokuments befinden und größer als die des vorherigens Kapiteleintrags sein, da ein IHV den chronologischen Aufbau des Dokuments beschreibt.

Beispielsweise erfüllen Einträge wie „Introduction 5“ oder „1. Introduction…5“ alle Kriterien. Im Gegensatz dazu erfüllt weder die Überschrift „1. Introduction“ aufgrund fehlender Seitenzahl noch der Produktname „Optiflux 1000“ aufgrund der hohen Zahl außerhalb des PDFs alle Kriterien. Auch ein Eintrag wie „1. Optiflux 10“, welcher wie ein typischer IHV-Eintrag aussieht, wird nicht als solcher erkannt, wenn er beispielsweise auf den Eintrag „2. Instructions…11“ mit einer höheren Seitenzahl folgt.

10 Zuordnung von Text zu Kapiteln

Um Servicetechnikern die passenden Textabschnitte zurückgeben zu können ist es essentiell, dass jeder Textabschnitt im Gesamtkontext des PDF-Dokuments eingeordnet werden kann. Um dies zu ermöglichen, müssen wir für jeden extrahierten Texteintrag das dazugehörige Kapitel bestimmen.

ANSPRECHPARTNER:
- **Dr. Robert Pesch (inovex, rpesch@inovex.de)**
- **Dr. Martin Krawczyk-Becker (KROHNE Messtechnik, M.Krawczyk-Becker@KROHNE.com)**

Alle Textelemente, die vor dem Inhaltsverzeichnis (IHV) sind, weisen wir dem Kapitel “Einleitung” zu. Einträge im IHV und zwischen dem IHV und dem ersten weiteren Kapitel weisen wir das Kapitel “Inhaltsverzeichnis” zu. Falls das Textelement weder zur Einleitung noch zum IHV gehört, wird zunächst eine Liste aller Kapitel erstellt, die auf der derzeitigen Seite beginnen. Dann berechnen wir die Bigram-Ähnlichkeit zwischen dem Textelement und den Kapitelnamen und verwenden den chronologisch ersten Kapitelnamen mit einer Ähnlichkeit >= 0.6. Die Bigram-Ähnlichkeit gibt in unserem Fall an, wie viele Bigramme (Sequenzen von zwei Wörtern) übereinstimmen. Falls die Ähnlichkeit hoch ist, ist die Wahrscheinlichkeit sehr hoch, dass es sich bei dem Textelement um den Kapitelnamen und somit den Beginn eines neuen Kapitels handelt. Wenn ein Textelement im späteren Teil des PDFs nicht selbst eine Kapitelüberschrift ist, wird das zuletzt gefundene Kapitel verwendet, da sich das Textelement in diesem befindet. Eine Übersicht befindet sich in Abb. 3.

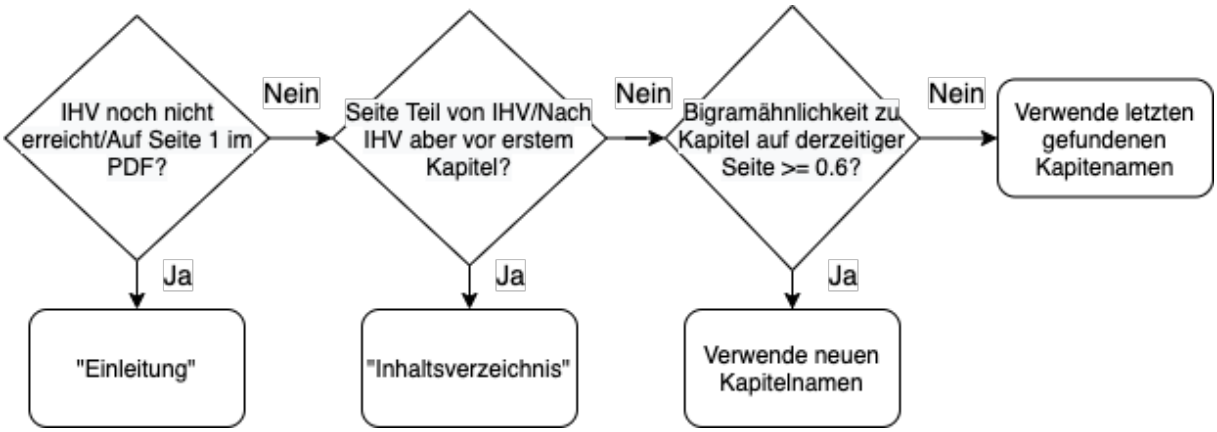


Abb. 3: Workflow zur Zuordnung von Text zu Kapiteln

11 Repräsentation der PDF-Dokumente im JSON-Format

Nach der Extraktion werden die Elemente als JSON-Objekt gespeichert, transformiert und in Elasticsearch importiert. Ein Textelement beinhaltet dabei immer den Text eines ganzes Kapitels. Ein beispielhaftes JSON-Objekt sieht wie folgt aus:

ANSPRECHPARTNER:
- Dr. Robert Pesch (inovex, rpesch@inovex.de)
- Dr. Martin Krawczyk-Becker (KROHNE Messtechnik, M.Krawczyk-Becker@KROHNE.com)

```
[
{
  "section": "Introduction",
  "content": "1 SAFETY INSTRUCTIONS OPTIFLUX 1000 1.3.5  Warnings and
symbols used Safety warnings are indicated by the following symbols. DANGER!
This information refers to the immediate danger when working with electricity",
          "uri" : "...",
  "file_name": "OPTIFLUX_1000_Handbook_en.pdf",
  "type": "Handbook",
  "language": "en",
  "device" : "OPTIFLUX 1000",
  "pages": [8],
  "images":
  [
        {
                  "base64": "iVBORw0KG...",
                  "embedding": [        5.934669494628906, 20.147275924682617,
7.831840515136719],
                  "page": 8
  }
]
}
]
```


12 Evaluation der Ergebnisse

Bei der Evaluierung der Ergebnisse ist es wichtig, sowohl die inhaltliche Korrektheit als auch die Reihenfolge der extrahierten Elemente zu überprüfen.

Um die Performance unserer Extraktion einschätzen zu können ist es notwendig zu wissen, wie eine komplett korrekte Extraktion aussehen würde. Hierfür haben wir sowohl für die reine Textextraktion als auch für die Inhaltsverzeichnisextraktion einige Seiten manuell aus den PDF-Dokumenten kopiert. Um die Einträge miteinander vergleichen zu können, weisen wir jedem mithilfe unseres Codes extrahierten Eintrag einen manuell extrahierten, korrekten Eintrag zu

Dies machen wir erneut unter Verwendung der Bigram-Ähnlichkeit. Dazu weisen wir jedem extrahierten Eintrag den ähnlichsten korrekten zu, wobei darauf geachtet wird, dass kein Eintrag doppelt zugewiesen wird. Zur Evaluation der inhaltlichen Korrektheit iterieren wir über alle extrahierten Sätze und berechnen den

ANSPRECHPARTNER:
- Dr. Robert Pesch (inovex, rpesch@inovex.de)
- Dr. Martin Krawczyk-Becker (KROHNE Messtechnik, M.Krawczyk-Becker@KROHNE.com)

Durchschnitt aller Bigram-Ähnlichkeitswerte. Dadurch erhalten wir einen Wert zwischen 0 und 1 der angibt, wie viele der Bigramme im Durchschnitt übereinstimmen.

Zur Evaluation der Reihenfolge iterieren wir über alle extrahierten Sätze und vergleichen den Satz mit allen anderen extrahierten Sätzen. Hierbei können folgende zwei Fehler auftreten:

1. Fehler: Der Index eines extrahierter Satz ist niedriger als der jetzige aber der Index des zugehörigen korrekten Satzes ist höher.

2. Fehler: Der Index eines extrahierter Satz ist höher als der jetzige aber der Index des dazugehörigen korrekten Satzes ist niedriger.

Nun berechnen wir, in wie vielen unserer Vergleiche kein Fehler auftrat und bekommen ebenso einen Wert zwischen 0 und 1, der uns ein Maß für die Korrektheit der Reihenfolge gibt. Diese etwas komplexere Fehlerberechnung basierend auf Vergleichen ist notwendig, da ein Vergleich basierend auf der absoluten Indexposition in diesem Fall nicht aussagekräftig ist. Der Grund hierfür ist, dass die Anzahl der extrahierten und tatsächlich existierenden Elemente meist ungleich ist. Bei einer Verschiebung der Elemente um eine Position würde hier bereits kein Index mehr übereinstimmen.

Bei unseren Experimenten ergaben sich für Text- und Inhaltsverzeichnisextraktion sowohl bei der Korrektheit des Inhalts als auch der Reihenfolge jeweils Werte zwischen 0.8 und 0.9.

Fazit

Die Extraktion von Text, Bildern und Tabellen war aufgrund der Schwierigkeiten, die das Portable Document Format mit sich bringt alles andere als einfach und funktionierte auch mit geeigneten Bibliotheken nicht komplett fehlerfrei. Mithilfe einiger eigener Ansätze gelang es dennoch, sämtliche relevante Inhalte aus den PDF-Dokumenten zu extrahieren und deren logische Struktur zu erkennen.

In der Zukunft wollen wir die extrahierten Tabellen ebenfalls in einem geeigneten JSON-Schema ablegen und in Elasticsearch importieren. Hierfür müssen die Tabellen genauso wie Text und Bilder ihrem jeweiligen Kapiteln zugeordnet werden. Zur Erkennung von Vektorgrafiken wollen wir einen eigenen Ansatz implementieren, welcher die einzelnen PDF-Seiten zunächst zu SVG konvertiert um daraufhin die gesuchten Bilder identifizieren und extrahieren zu können.

Außerdem wollen wir die extrahierten Bilder und Bild-Embeddings zur Produktidentifikation nutzen. Wir erlauben es Servicetechnikern, Produktbilder von ihrem Einsatz hochzuladen, woraufhin in Elasticsearch basierend auf den Bild-Embeddings das ähnlichste Bild identifiziert wird. Funktioniert die Suche, handelt es sich bei dem ähnlichste Bild um dasselbe Produkt. Nach der Identifikation können zusätzliche relevante Daten aus den PDF-Dokumenten zur Unterstützung des Servicetechnikers zurückgegeben werden.

ANSPRECHPARTNER:
- Dr. Robert Pesch (inovex, rpesch@inovex.de)
- Dr. Martin Krawczyk-Becker (KROHNE Messtechnik, M.Krawczyk-Becker@KROHNE.com)