Richtlinie zum „Software-Sprint“

doc2data

Schlussbericht

Zuwendungsempfänger:

Sergej Levich

Das diesem Bericht zugrundeliegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 01IS22S04 gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt beim Autor.

Kurze Darstellung der Aufgabenstellung und Motivation

Die Verarbeitung von Dokumenten ist eine allgegenwärtige Aufgabe. Insbesondere in Verwaltungsprozessen wird diese hauptsächlich manuell ausgeführt, was zu hohen Kosten, Verzögerungen und teilweise Intransparenz führt. Covid hat die Problematik der Überlastung der Behörden aufgrund operativer Ineffizienz deutlich gemacht.

Mit doc2data verfolgte der Autor das Ziel, eine Software zur Beschleunigung und Vereinfachung der Dokumentenverarbeitung zu entwickeln. Konkret resultierte das Vorhaben in einer Python-Bibliothek, welche das Anlernen von Machine Learning (ML) Modellen für eine Reihe von Aufgaben ermöglicht. Damit können Dokumente in Teilen automatisiert verarbeitet werden.

Das ursprüngliche Vorgehen orientierte sich an einer Konzeptphase mit initialem Nutzerfeedback und einer anschließenden Entwicklungsphase. Vor Projektbeginn konnten jedoch zwei Kooperationspartner akquiriert werden, welche Dokumentendaten aus realen Prozessen zur Verfügung gestellt haben. Damit konnte die Entwicklung agil entlang echter und gängiger Problemstellungen durchgeführt werden. Folglich wurde mit doc2data eine eigenständige Bibliothek entwickelt und zeitgleich in der Praxis pilotiert.

Wichtige Meilensteine waren hierbei:

- Recherche (Parsing von PDFs, Deep Learning Architekturen)
- Entwicklung eines Moduls für die Erstellung und Verwaltung von PDF-Sammlungen
- Durchführung des Modelltrainings für vier Aufgaben der Dokumentenverarbeitung
- Entwicklung von Modulen für weitere Teile der Trainingspipeline
- Erstellung der Dokumentation

Durch die Beteiligung von Kooperationspartnern sind zusätzliche administrative Aufgaben bearbeitet worden. Dazu gehörten Abstimmungen, die Zusammenarbeit beim Aufsetzen der IT-Infrastruktur sowie die Durchführung von (Abschluss-)präsentationen.

Beitrag des Projektes zu den Zielen der Förderinitative „Software-Sprint“

Das Projekt doc2data kann als Softwareinfrastrukturprojekt verstanden werden, da es nicht auf Endnutzer, sondern auf Entwickler*innen als Zielgruppe ausgerichtet ist.

Die Entwicklung eigener ML-Modelle ist für Entwickler*innen ohne entsprechende Erfahrung nur schwer möglich. doc2data verbindet eine Vielzahl bestehender Python Bibliotheken um eine klar strukturierte und einfach zu nutzende Pipeline für das Modelltraining bereitzustellen. Damit können Entwickler*innen in wenigen Schritten ML-Modelle für die Dokumentenverarbeitung auf eigenen Daten anlernen.

Auf Basis dieser Modelle kann in der Folge eine Vielzahl von Prozessen effizienter gestaltet werden. Im Bereich der öffentlichen Verwaltung betrifft dies hauptsächlich administrative Prozesse, die Anträge auf Basis eingereichter Dokumente abwickeln. Beispiele sind die Verschlagwortung (Klassifikation) von Dokumenten im Posteingang, die Zuordnung von Dokumenten zu Akten oder das automatisierte Auslesen von Feldern aus nicht standardisierten Formularen.

Im Projektverlauf wurden neben den Verwaltungen auch Civic-Tech Entwickler*innen als Zielgruppe identifiziert (z.B. FragDenStaat, investigative Journalist*innen). Hier kann doc2data eingesetzt werden um große, bestehende Dokumentensammlungen zu strukturieren und zu analysieren. So kann eine kleine Anzahl von Dokumenten nach einem bestimmten Kriterium manuell annotiert werden, um anschließend den Rest der Sammlung durch das ML-Modell zu bewerten.

Die ML-basierte Dokumentenverarbeitung (Document AI) wird aktuell durch den kommerziellen Markt dominiert. Open-Source-Entwicklungen in diesem Bereich bieten kleineren bzw. finanzschwachen Organisationen einen Weg, von den neusten technologischen Entwicklungen zu profitieren.

Ausführliche Darstellung der Ergebnisse

Aufgrund der agilen Entwicklung waren der Ressourceneinsatz fix und die Meilensteine variabel. Diese wurden im Verlauf des Projekts in insgesamt zehn Reviews angepasst. Dabei flossen die Beobachtungen und Herausforderungen aus der Arbeit mit realen Dokumentendaten in die Definition und Priorisierung der Meilensteine ein. Entsprechend wurde beispielsweise die Entwicklung eines UI zur Visualisierung der Ergebnisse zugunsten eines Annotationstools zurückgestellt. Ziel bei solchen Entscheidungen war stets, die Features so zu definieren, dass die spezifischen Probleme gelöst werden konnten, ohne dass die Bibliothek an Generalisierbarkeit einbüßt.

Insgesamt können die Ergebnisse aus vier Perspektiven betrachtet werden.

1 Funktionalität

Mit doc2data können Entwickler*innen auf Basis von Dokumenten im PDF-Format und entsprechenden Annotationen ML-Modelle für die folgenden vier Aufgaben anlernen:

1. Rotation von einzelnen Dokumentenseiten
2. Zuschneiden von einzelnen Dokumentenseiten
3. Klassifikation von Textelementen innerhalb einer Dokumentenseite
4. Klassifikation von mehrseitigen Dokumenten als Ganzes

Mit dem Annotationstool doc2data UI können Entwickler*innen Textelemente einzelner Dokumentenseiten annotieren um im nächsten Schritt mit doc2data Modelle für die Klassifikation von Textelementen (3) anzulernen.

2 Module der Bibliothek

doc2data besteht aus einer Anzahl von Python-Modulen, die Funktionalität für einzelne Modellierungsschritte bündeln. Im Wesentlichen können diese entlang einer sequenziellen Anwendung wie folgt dargestellt werden:

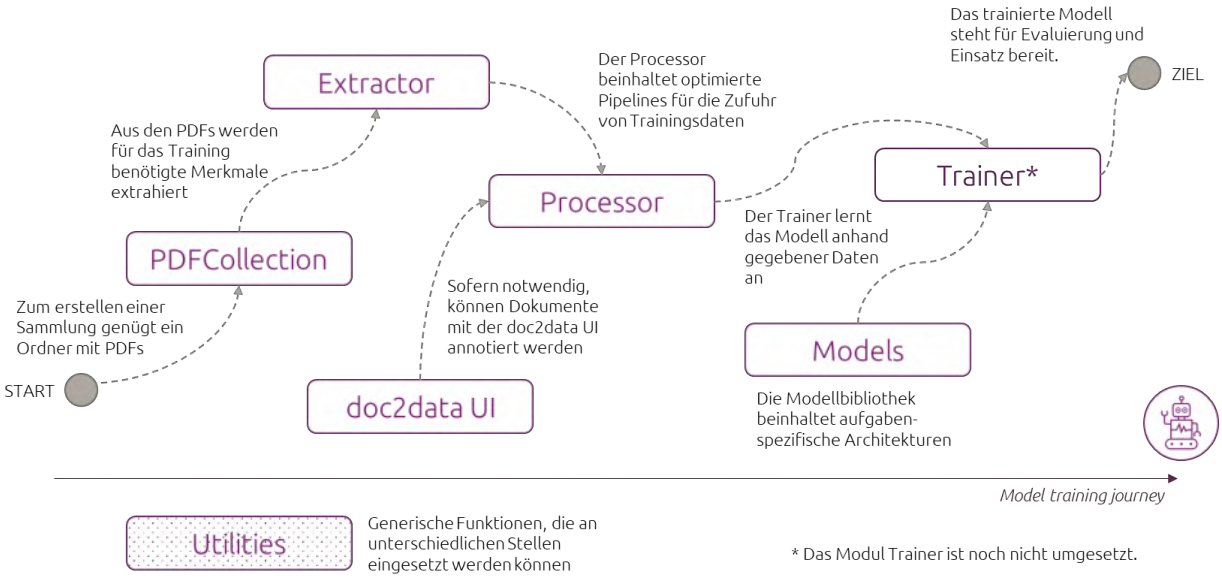


Abbildung 1. Übersicht der Module: Vom PDF zum KI-Modell mit doc2data.

Die vier verschiedenen Modellierungsaufgaben (siehe Funktionalität) werden in den Modulen *Processor* und *Models* eigens abgebildet. Alle anderen Module funktionieren unabhängig von der gewählten Aufgabe.Das Annotationstool doc2data UI ist als separate, Python-basierte Applikation konzipiert, die jedoch auf doc2data aufbaut.

Technologie

doc2data basiert auf gängigen Open-Source-Bibliotheken, die hauptsächlich über Python-Schnittstellen eingebunden werden. Wesentliche Komponenten sind nachfolgend dargestellt:

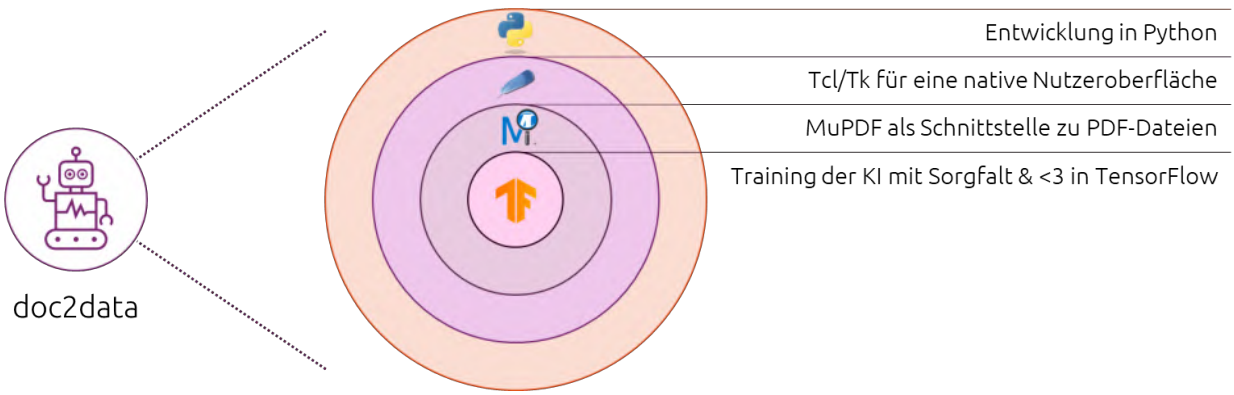


Abbildung 2. Technologische Komponenten von doc2data

Die Dokumentation basiert auf den Bibliotheken *MkDocs* und *mkdocstrings* und ist auf *Read the Docs* gehostet.

Anwendungsfälle

Im Rahmen der Zusammenarbeit mit den erwähnten Kooperationspartnern wurde doc2data in zwei Projekten pilotiert:

1. Klassifikation von Dokumenten (Anwendungsfall bei einer Berufsgenossenschaft)
2. Verarbeitung von Kfz-Scheinen (Anwendungsfall bei einem Automobilclub)

Der erste Anwendungsfall ist im Posteingang angesiedelt und ist typisch für viele Behörden. Ziel ist hierbei die Zuordnung einer vorab definierten Kategorie für eingegangene Dokumente. Gängige Kategorien sind z.B. Rechnung, Arztbericht, Laborergebnis oder Anschreiben. Im Hinblick auf die Funktionalität von doc2data kam hier das Modell für die Klassifikation von mehrseitigen Dokumenten (4) zum Einsatz.

Der zweite Anwendungsfall beschäftigt sich mit der Extraktion spezifischer, vorher definierter Felder aus Dokumenten – einer weit verbreiteten Aufgabenstellung. Konkret ging es um die Ermittlung des Kennzeichens, des Fahrzeughalters sowie der Fahrzeugspezifika aus Kfz-Scheinen. Da Kfz-Scheine als Fotos/Scans vorlagen, musste eine Vorverarbeitung stattfinden. Damit kamen im Hinblick auf die Funktionalität von doc2data die Modelle für die Rotation (1), das Zuschneiden (2) sowie die Klassifikation von Textelementen (3) einer Dokumentenseite zum Einsatz.

Im Hinblick auf die Erkenntnisse aus der Projektarbeit wurde der Autor in seiner Einschätzung der Notwendigkeit und des Bedarfs nach Open-Source-Lösungen für die Dokumentenverarbeitung bestärkt. Auch sind die Herausforderungen und Chancen in der Zusammenarbeit mit staatlichen Akteuren deutlich geworden. Die Begleitung durch die Open Knowledge Foundation und den Prototypefund war insbesondere durch die Vernetzung mit der Civic Tech Community und potenziellen Nutzer*innen höchst wertvoll.

Zielgruppe, Nutzen und mögliche Weiterentwicklungen

Der zentrale Nutzen von doc2data besteht im einfachen Zugang zu ML-Modellen für die Dokumentenverarbeitung. In der Folge lassen sich Problemstellungen in Prozessen und Projekten effizienter lösen, als es durch eine manuelle Bearbeitung möglich wäre. Während die primäre Zielgruppe Behörden und freie Entwickler*innen umfasst, ist das Thema Dokumentenverarbeitung so allgegenwärtig, dass ein deutlich größerer Anwenderkreis denkbar ist. Von Forschungseinrichtungen, die auf Datenbeschaffung aus großen Dokumentensammlungen angewiesen sind, zu KMUs, die operative Prozesse abwickeln - die Breite an kommerziellen und nicht-kommerziellen Anwendungsfällen ist signifikant.

Gegeben dessen, dass die Verarbeitung natürlicher Sprache[1] aktuell eine der bedeutendsten Stoßrichtungen im Bereich Künstlicher Intelligenz darstellt, ist hier in der Zukunft mit deutlichen technologischen Sprüngen zu rechnen. Projekte wie doc2data können dazu beitragen, die Lücke zwischen Forschung und Anwendung an großen kommerziellen Akteuren vorbei in einer transparenten Art und Weise zu schließen.

[1] Natural Language Processing (NLP)

Vor diesem Hintergrund soll doc2data weiterhin an der Schnittstelle von Wissenschaft und Anwendung entwickelt werden. Es werden Kooperationsmöglichkeiten mit Forschungsinstitutionen[2] ausgelotet. Ferner werden Möglichkeiten zur Monetarisierung von doc2data durch bspw. komplementäre Dienstleistungen für größere Organisationen untersucht.

Die Arbeit an doc2data hat den Autor auch im Hinblick auf die persönliche und fachliche Weiterentwicklung weitergebracht. Besonders hervorzuheben ist das gewonnene Wissen rund um die professionelle Entwicklung von Open-Source-Software in Python. Der Prototypefund bietet aus der Sicht des Autors eine einmalige Gelegenheit, ein relevantes Problem mit einem ausschließlichen Fokus auf den Mehrwert der Lösung zu adressieren.

Kurze Darstellung der Arbeiten, die zu keiner Lösung geführt haben

In der Modellierung mit realen Daten wurden unterschiedliche Modellarchitekturen sowie Parameterkombinationen evaluiert. Entsprechend wurden nur die besten Experimente beibehalten und der Rest verworfen. Für die Aufgaben (1), (2) und (3) mussten Dokumente manuell annotiert werden, sodass der Datensatz nur wenige hundert Beispiele umfasst. Damit konnte keine hinreichend hohe Modellgenauigkeit erreicht werden. Daraufhin mussten Methoden der „data augmentation“, d.h. der künstlichen Anreicherung der Daten, eingesetzt werden, um die Qualität der Modelle zu erhöhen.

Kurze Angabe von Präsentationsmöglichkeiten für mögliche Nutzer

Der Quellcode des Projekts ist auf zwei Repositorien aufgeteilt (doc2data & doc2data UI).

doc2data (Bibliothek): https://github.com/serge724/doc2data

doc2data UI (Annotations-Tool): https://github.com/serge724/doc2data-ui

Dokumentation für doc2data: https://doc2data.readthedocs.io/en/latest/

Webseite des Anwendungsfalls (1): https://sigai.de/featured/theses/thesis2/

Kurze Erläuterung zur Einhaltung der Arbeits- und Kostenplanung

Das Projekt wurde agil durchgeführt, sodass die Planung in regelmäßigen Abständen angepasst wurde. Aufgaben, die größere Aufwände als geplant verursachten, betrafen das Einhalten von Standards in der Entwicklung, die Kommunikation mit den Kooperationspartnern sowie die Entwicklung des Annotations-Tools (doc2data UI). Teilaufgaben mit Minderaufwänden gab es dagegen bei der Implementierung der PDF-Schnittstelle sowie des vortrainierten Modells *LayoutXLM*.

Ein neuer Release des Open-Source-Tools *PaddleOCR* war inkompatibel mit den Abhängigkeiten von doc2data. Es konnte auf die Alternative d*ocTR* ausgewichen werden, was sogar zu einer höheren Qualität der Zeichenerkennung führte.

[2] ZHAW Zürcher Hochschule für Angewandte Wissenschaften, Institut für Medizinische Informatik der Charité

Kurze Darstellung von etwaigen Ergebnissen bei anderen Stellen

Während die meisten Modellarchitekturen vom Autor selbst aufgesetzt wurden, erfolgte die Integration des vortrainierten Modells *LayoutXLM* durch die Open-Source-Bibliothek *huggingface* (https://github.com/huggingface), sodass keine eigene Implementierung notwendig war. Zusätzlich ist im Entwicklungszeitraum die OCR Bibliothek *docTR* (https://github.com/mindee/doctr) verfügbar geworden.