Umwandlung von Text zu XML mit Python, auf Basis des Digitalisates „Bomber's Baedeker"

Felix Bach und Cristian Secco

Ziel dieses Projektes war es, Strukturen innerhalb eines digitalisierten Textes zu erkennen und diese automatisch zu extrahieren und in einem XML-Format zu speichern. Die verwendeten Daten stammten aus dem Digitalisat des „Bomber's Baedeker", eine Sammlung von deutschen Infrastrukturinformationen für Bomberpiloten der britischen Luftwaffe aus dem Zweiten Weltkrieg. Diese konnten dann mithilfe von Skripten in der Programmiersprache Python untersucht werden. Die per Hand definierten Textabschnitte konnten so in dem kompletten Werk automatisch erkannt werden. Anschließend wurden diese Informationen in XML transformiert, um die Struktur des Originals zu repräsentieren, aber auch um eine Möglichkeit zu schaffen, die kompletten Daten schnell zu durchsuchen und mögliche Verbindungen innerhalb dieser aufzuzeigen.

Einleitung

Dieses Projekt entstand im Rahmen des Seminars „Digitalisierung: Grundlagen und Praxis" innerhalb des Bachelor-Studienganges „Informationswissenschaft" an der Hochschule Darmstadt. Leitende Dozenten des Seminars waren Prof. Dr. Elke Lang und Prof. Dr. Stefan Schmunk. Die inhaltliche Zielsetzung des Projektseminars war es, einen praktischen Einstieg in die verschiedenen Aspekte der Digitalisierung zu ermöglichen sowie die diversen Anwendungsbereiche dieser zu verdeutlichen. Die zentrale Frage unseres Projektes war, wie man mithilfe von Algorithmen ein Verfahren entwickeln kann, welches Muster in Texten erkennt und diese automatisch in einer strukturierten Form speichert.

Kurz zusammengefasst, wurde der Baedeker von der Royal Air Force genutzt, um während des Zweiten Weltkrieges deutsche Industriestandorte anzugreifen. Das Buch besteht aus zwei Bänden und verfügt über Informationen zu geografischer Lage, Einwohnerzahl, Entfernung zu London in Meilen sowie einer umfassenden Beschreibung von mehr als 500 Städten in Deutschland.

In diesem Beitrag präsentieren wir einen Ansatz, welcher es erlaubt, gedruckte Werke in eine XML-Datei umzuwandeln. Der Vorgang erfolgte anhand eines Python-Skripts am Beispiel des gut strukturierten Werkes Bomber's Baedeker, das physisch und digitalisiert der Universitätsbibliothek Mainz vorliegt.

Zielsetzung

Unser Ziel war es, ein Programm zu entwickeln, welches auf einer theoretischen Ebene zeigt, dass eine automatische Strukturerkennung innerhalb eines unstrukturierten Digitalisats möglich ist und diese zu einem Informationsgewinn führen kann. Insofern handelt es sich bei unserem Projekt um eine Machbarkeitsstudie und das Ziel war es, mit gängigen Erschließungs- und Aufzeichnungsverfahren zu eruieren, mit welchem zeitlichen Aufwand diese angewandt werden können. Dies ist insofern von großer Relevanz, als dass zwar größere Bestände mittlerweile retrodigitalisiert vorliegen und deren Inhalte auch oftmals mittels einer OCR-Software in Volltexte transformiert wurden, aber diese aufgrund der Fehlerraten der nicht identifizierten Layoutelemente etc. für maschinelle Analyseverfahren weiter aufbereitet werden müssen.

Vorgang

Um diesen Beweis zu erbringen, haben wir zuerst die analoge Fassung des Bomber's Baedeker analysiert, um manuell eine Einteilung in Strukturen vorzunehmen, die wir dann mit der Programmiersprache Python automatisch extrahieren können.

Inhalte des Druckwerkes werden stark strukturiert dargestellt und sind daher sehr gut geeignet, um enthaltene Informationen in eine maschinenlesbare, objektorientierte und strukturierte Form zu bringen. An dieser Überlegung knüpft das Projekt an. Wir versuchen, einen automatisierten Prozess zu erstellen, bei dem das Schriftstück zu einem XML-Dokument umgewandelt wird, welches dann in weiteren Schritten für eine Weiterverarbeitung oder Visualisierung genutzt werden kann.

Das auf Webpage der Universitätsbibliothek Mainz zur Verfügung stehende Digitalisat liegt als PDF-Datei vor und kann kostenlos heruntergeladen werden. Das Öffnen der heruntergeladenen Datei zeigte, dass das Dokument bereits durch eine OCR verarbeitet wurde. Jede Seite verfügt über eine Text-Layer, die es ermöglicht, im Dokument zu suchen. Wie sich im weiteren Verlauf des Projektes herausgestellt hat, sind die Ergebnisse der verwendeten OCR aber nur für eine

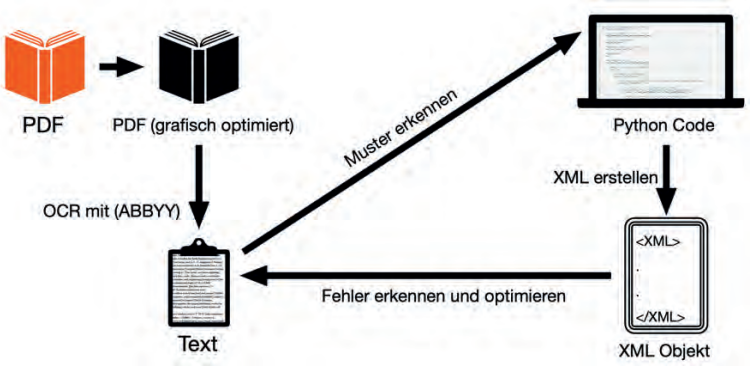

Abb. 1: Workflow zur Verbesserung der Ergebnisse

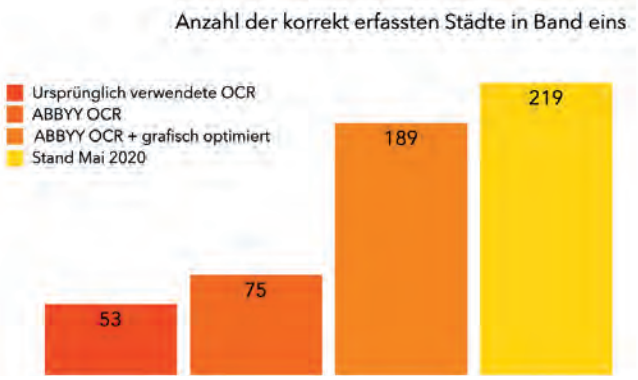

Abb. 2: Von Python-Code gefundene Städte

einfache Suche verwendbar und nicht für eine direkte Umwandlung geeignet.

Unser geplanter Workflow bestand daher darin, Regeln für Strukturen manuell zu definieren, Algorithmen zu schreiben, die diese Definitionen erkennen, Daten zu suchen, diese Daten zu speichern und letztendlich die Ergebnisse auszuwerten. Leider mussten wir feststellen, dass die OCR (Optical Character Recognition)-Qualität des uns zur Verfügung gestellten Digitalisats viele Fehler enthielt.

Anhand einer sehr kleinen erzeugten Ergebnismenge im anfänglichen Versuch kristallisierte sich eine von uns nicht eingeplante Schwierigkeit heraus, welche die Idee des Projektes infrage stellte: Wir erkannten schnell, dass die von der Universitätsbibliothek Mainz verwendete OCR nicht zuverlässig arbeitet. Nach eigener Schätzung sowie einer Verrechnung der erhaltenen Ergebnisse (53) mit der Sollergebniszahl von 240 bis 260, kamen wir auf eine OCR-Genauigkeit von 20 bis 30 %. Der von uns erwartete Wert müsste mindestens bei 95 % liegen.

Viel versprechendere Ergebnisse wurden mit einer OCR-Software von ABBYY erreicht, welche im weiteren Verlauf verwendet wurde. Um ein möglichst genaues Arbeiten der Software zu ermöglichen, ist hier eine Segmentierung nötig gewesen; eine eindeutige Abgrenzung zwischen dem relevanten Text und irrelevantem Hintergrund. Wie bereits erwähnt, ist diese Abgrenzung im Baedeker nicht immer genau möglich. Eine nachträgliche adaptive Banalisierung wurde daher vorgenommen und in unserem Beispiel von der Adobe Software „Photoshop“ und im weiteren Verlauf von der Apple Mac internen Software „Fotos“ ausgeführt. Im weiteren Verlauf wurden einige Versuche unternommen, um die Ergebnismenge zu steigern.

Zur weiteren Steigerung der Ergebnismenge mussten wir einen Prozess der Fehlerfindung und Python Code Anpassung, sprich Preprocessing vornehmen. Der Zyklus bestand aus vier Kontrolldurchläufen, die größtenteils eine Verbesserung von einzelnen, von der OCR falsch aufgefassten Zeichen beinhaltet.

Beispiele für von uns verbesserte Zeichen sind: (0 - O), (rn - m), (* - °), (; - :), (B - E), (F - E)

Nach dem letzten Durchlauf erhielten wir eine Ergebnismenge von 219 Städten, also eine Genauigkeit von 88 %. Dieses Ergebnis besteht dennoch nur aus den Städten, sowie Bundesländern, Koordinaten, Entfernung zu London und Einwohnerzahl, welche im Druckwerk erwähnt werden. Der restliche Text (die detaillierten Auflistungen der Ziele) besitzt nur noch eine Genauigkeit von 50 %. Um das Gesamtprodukt der OCR zu verbessern, wäre es nötig, das originale Druckwerk nochmals zu scannen. Nach der gezeigten Untersuchung, halten wir es für notwendig, die Belichtung des Druckwerkes bei der Digitalisierung zu verbessern und eine Aufnahme mit höherer Auflösung zu erstellen.

Die Aufbereitung des Digitalisates, die Verwendung der ABBYY-OCR und das Preprocessing sind wichtige Faktoren, um die Ergebnismenge zu steigern. Wie in der Abbildung zu sehen, konnten wir die Ergebnismenge durch diese Maßnahmen von 53 auf 219 erhöhen.

Der folgende Abschnitt des Beitrags beschreibt, die von uns verwendete technische Anwendung von Python, zur Erfassung von Strings mit erwähnten Kriterien sowie der Ausgabe und Export der Ergebnisse als XML und CSV Objekt.

Eine automatisierte XML-Erschließung mithilfe von Python stellte sich als überraschend einfach heraus. Für das Dokument relevante Muster ließen sich meist relativ genau in Programmiersprache umwandeln und garantierten eine strukturierte Umwandlung. Die Struktur des finalen Python-Skripts kann in drei Ebenen unterteilt werden. Ebene eins stellt das Preprocessing dar. Es werden wiederholende Fehler der OCR und der Verfasser verbessert, die im weiteren Verlauf des Skriptes zu Fehlern oder falscher Erfassung der Objekte führen könnten. Zusätzlich werden strukturelle Abweichungen, wie beispielsweise die Benutzung eines Semikolons statt Doppelpunkten an im Baedeker größtenteils benutzte Normen angepasst.

Die zweite Ebene beinhaltet die Objekterkennung und -erfassung. Durch von uns festgelegte Muster erkennt das Skript, wie der Name einer Stadt im Baedeker aussieht und welche Attribute die Stadt besitzt.
Städtenamen werden in Großbuchstaben geschrieben. Nach einem Städtenamen folgt in Klammern das Bundesland, die Provinz oder Region der Stadt. In der nächsten Zeile werden Koordinaten der Stadt im Format „00° 00' N. 00° 00' E: „angegeben. Diese können aber variieren und auch nur einstellig sein, beziehungsweise auch keine Nachkomma-Koordinate haben, zum Beispiel „00° N. 00° E:“ Auf die Koordinaten folgt die Entfernung zu London in Meilen im Format „000 miles:“ Die letzte Information der Kopfzeile gibt die Einwohnerzahl in Klammern an.
Eine Schleife sucht im gesamten Text nach den von uns definierten Mustern und sichert diese Struktur in der nächsten Ebene als XML. In der letzten Ebene werden die gefundenen Städte als Objekt in der XML abgelegt und zusätzlich mit den dazugehörigen Attributen versehen.

Resultat

Als Ergebnis entstand so eine XML-Datei, die alle Städtenamen mit den jeweils dazugehörigen Informationen (Region, Koordinaten, Entfernung zu London und Einwohnerzahlen) enthält. Um diese zu extrahieren, war es am Ende nur notwendig, den Text einzulesen und das Programm auszuführen.
Die von uns erstellte XML kann beispielsweise zur Datenanalyse verwendet werden. Die folgende Abbildung zeigt eine Karte von Deutschland mit einigen Punkten. Blaue Punkte zeigen Koordinaten, die im Baedeker aufgeführt werden. Rote Punkte zeigen die aktuellen Koordinaten der Städte. Es ist zu sehen, dass viele Koordinaten von den Autoren falsch aufgefasst wurden.
Dieses Experiment diente dazu, eine der Möglichkeiten aufzuzeigen, wie digitalisierte Daten verarbeitet werden können, um aus diesen maschinelle und interpretierbare Daten zu erzeugen. In diesem Fall konnten wir die Koordinaten aus dem Originaltext auf einer Karte visualisieren, um einen Vergleich mit der tatsächlichen Lage der Städte durchzuführen. Dies ist allerdings nur ein Beispiel. Durch die Umwandlung von einer Text-Datei in eine strukturierte XML-Datei können mit allen erhaltenen Informationen Vergleiche oder weitere Analysen durchgeführt werden.
Selbst wenn es uns nicht möglich war, sämtliche Informationen aus dem Druckwerk zu extrahieren, sind die von uns entnommenen Daten hilfreich für eine Weiterverarbeitung und Visualisierung. Demnach konnten wir das Ziel unserer Idee verwirklichen, wenn auch nur bedingt. Um ein besseres Ergebnis erhalten

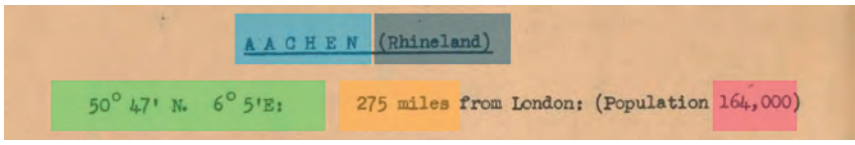

Abb. 3: Festgelegte Muster am Beispiel Aachen
https://visualcollections.ub.uni-mainz.de/histbuch/content/pageview/454135

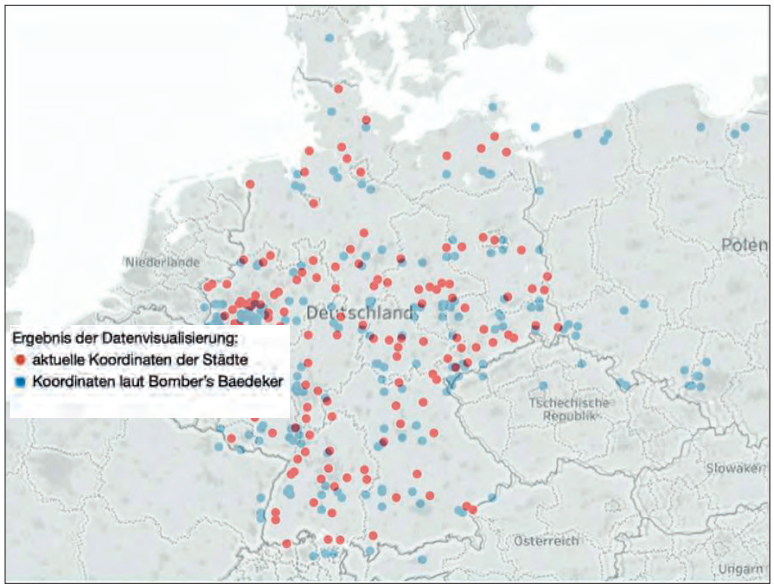

Abb. 4: Gegenüberstellung der Koordinaten, erstellt mit Tableau: https://www.tableau.com/de-de

zu können, würden wir eine neue Digitalisierung mit verbesserten Bedingungen vorsehen.

Future Work

Das dargestellte Skript zeigt eine mögliche Lösung, um automatisch XML-Dateien aus gedruckten Werken zu erzeugen. Um unseren Ansatz noch zu Erweitern und zu verbessern, sehen wir es vor, beispielsweise Machine-Learning-Verfahren zu verwenden, um Fehler im Text zu erkennen sowie eine akkurate Mustererkennung zu gewährleisten. Letzteres ist vor allem wichtig, um weitere Werke automatisch umwandeln zu können. Im Rahmen weiterer Forschung mit dem Digital Humanities Lab des Leibniz-Instituts für Europäische Geschichte in Mainz konnten wir unsere Ergebnisse verbessern und um weitere Informationen erweitern. Alle Städte der beiden Bände können extrahiert werden. Zusätzlich werden nun Informationen zu Industrie und Infrastruktur erkannt und ausgelesen. Der aktuelle Stand der Forschung ist in diesem Blog nachzulesen: https://dhlab.hypotheses.org/1820


Cristian Secco
cristian-s@hotmail.de
Felix Bach
fbach9310@gmail.com