

Kategorisierung von lokalisierten Bilddateien mittels OCR

Simon Gruber



BACHELORARBEIT (EXPOSÉ)

eingereicht am
Fachhochschul-Bachelorstudiengang

Software Engineering

in Hagenberg

im Juni 2023

Inhaltsverzeichnis

| | |
|--|-----------|
| Fragestellung | 1 |
| Forschungsstand | 2 |
| Problemstellung | 3 |
| 1 Herausforderungen | 3 |
| Vorläufige Gliederung | 5 |
| Zeitplan | 6 |
| Preprocessing (Probekapitel) | 7 |
| 2 Stand der Technik | 7 |
| 2.1 Optimierung | 8 |
| Postprocessing | 9 |
| 3 Stand der Technik | 9 |
| Literatur | 10 |
| 4 Optical Character Recognition | 10 |
| 5 Datenbankdesign und UML | 10 |
| 6 Approximate String Matching | 10 |
| Quellenverzeichnis | 11 |
| Literatur | 11 |

Fragestellung

Ziel dieser Bachelorarbeit ist es, herauszufinden, welche Vorgehensweisen bei der Texterkennung zu den besten Ergebnissen führen. Dazu werden die Ergebnisse, beeinflusst durch verschiedene Bildverarbeitungsschritte sowie Filtermethoden anhand gängiger Fehlermetriken für Texterkennungssysteme in einer prototypischen Implementierung verglichen.

Forschungsstand

Seit einigen Jahren wird optische Texterkennung in der Informationstechnik dazu verwendet, Texte in verschiedensten Grafiken als solche zu erkennen und zu extrahieren. In diesem Kontext sticht besonders die seit 2005 quelloffene „Tesseract OCR Engine“ [13] hervor, die mit mittlerweile über 50.000 Sternen auf der Website GitHub eine der größten OCR Engines darstellt.

Es gibt zahlreiche Werke, die sich mit der Funktionsweise von optischen Texterkennungswerkzeugen wie Tesseract befassen. Empfehlungen zur Verbesserung des zu verarbeitenden Ursprungsbildes wurden in der Literatur, wie auch in der Tesseract Dokumentation [13] bisher nur oberflächlich behandelt.

Für die Filterung der Ergebnisdaten werden unter anderem Vorgehensweisen aus dem gut erforschten Themengebiet des Natural Language Processing verwendet.

Problemstellung

Die in Salzburg ansässige COPA-DATA GmbH bietet die Softwareplattform zenon an, die als umfassende Gesamtlösung Unternehmen in zahlreichen Anwendungsgebieten bei der Automatisierung ihrer Herstellungsprozesse unterstützt.

Die zenon-Plattform kann sowohl vom Kunden selbst, als auch durch das Professional Services Team individuell auf Kundenanforderungen zugeschnitten und in bestehende Prozesse und vor allem Software eingebunden werden. Den Grundstein für die hohe Anpassbarkeit bildet die Produktdokumentation, in der Schnittstellendokumentation, Anleitungen und Beispiele in verschiedensten Sprachen, Formaten und mit kundenspezifischen Erweiterungen umfassend sowohl für Mitarbeiter, als auch für Kunden festgehalten sind.

In der Produktdokumentation werden, besonders in Hinblick auf die grafischen Tools wie die zenon Engineering Studio Entwicklungsumgebung oder die zenon Service Engine, zahlreiche Grafiken verwendet, um Beispiele verständlicher zu machen und Anleitungen übersichtlicher zu gestalten. Um bei dem großen Funktionsumfang der zenon-Tools, den vielen Sprachen, Anpassungen und den unterschiedlichen Themengebieten innerhalb der Dokumentation nicht den Überblick zu verlieren, benötigt das interne „Technical Content and Translation“ Team unterstützend zu dem intern verwendeten CMS „Author-It“ eine dedizierte Anwendung zur Verwaltung von sprachabhängigen Bilddateien.

Während das Programm auch die Basisfunktionalität, das effiziente Speichern, Bearbeiten, Löschen, Abrufen beziehungsweise das generelle Verwalten von Screenshots und der zugehörigen Metainformation abdecken soll, konzentriert sich diese Bachelorarbeit primär auf die Kategorisierungsfunktionalität.

Mithilfe von optischer Texterkennung (engl. optical character recognition, „OCR“) soll es den Mitarbeitern möglich gemacht werden, hochgeladene Screenshots und Grafiken innerhalb von kürzester Zeit aufgrund ihrer Inhalte zu verschlagworten und auf eine bestehende Liste von Schlagworten (engl. „Keywords“ beziehungsweise „Tags“) zu prüfen.

1 Herausforderungen

Die konkrete Herausforderung bei der Texterkennung mittels OCR besteht

Die konkrete Herausforderung bei der Texterkennung mittels OCR besteht aus dem Finden eines passenden Texterkennungs-Frameworks, der Einbindung der im Rahmen dieser Bachelorarbeit entwickelten Prototypbibliothek in das bestehende „Screenshot-Manager“ Basisprogramms und nicht zuletzt dem korrekten und zuverlässigen Erkennen der erwarteten bzw. bis dahin unbekannten Schlagworte in den bestehenden Bilddateien

und denen, die in Zukunft hochgeladen werden. Um bestmögliche Ergebnisse zu erzielen, ist das automatische Verändern der Bilddaten (Thresholds, Anpassen der Kontrastwerte, Farbe, Helligkeit, etc.) dabei unumgänglich.

Weiters bildet die Verknüpfung der Schlagworte mit den entsprechenden Screenshots unter Rücksichtnahme auf die Sprache der hochgeladenen Grafik die Basis des Schlagwortsystems. Es ist ein geeignetes Speichersystem notwendig, um bei der Vielzahl an gespeicherten Screenshots performant nach Schlagwörtern suchen zu können. Die OCR-Funktionalität und auch die Ablagestruktur der Schlagwortinformation muss modular aufgebaut sein, um zukünftig hinzukommende Dokumentationssprachen beziehungsweise neue Schlagwörter, beispielsweise bei Anpassung des Corporate Brandings, ohne großen Konfigurations- oder gar Entwicklungsaufwand unterstützen zu können.

Auch die Integration mit dem intern eingesetzten Content-Management-System (kurz „CMS“) „Author-It“ muss gegeben sein, wobei die Verwaltung und Verknüpfung der Bilddateien mit dem Author-It System aufwendiger ist, als die für die Bachelorarbeit relevante Ablage der Schlagwortinformation.

Vorläufige Gliederung

1. Einleitung, Aufgabenstellungen und Ziele
2. Themengrundlagen und Forschungsstand
3. Konzept / Design
 - 3.1. Texterkennung
 - 3.1.1. Vergleich verschiedener Bildtransformationstechniken
 - 3.1.2. Konzepte zur Filterung der Ergebnisdaten
 - 3.2. Datenbankstruktur
 - 3.2.1. Entwicklung einer Struktur für das Verbinden von Schlagwörtern mit Datenbankobjekten
 - 3.3. Programmaufbau
 - 3.3.1. Beschreibung der Bibliotheksarchitektur
 - 3.3.2. Schnittstellenbeschreibung
 - 3.3.3. Konzept für Vor- und Nachbearbeitung der Daten
 - 3.3.4. Entwurf der unscharfen Suche in den Ergebnisdaten
4. Prototypische Umsetzung
 - 4.1. Texterkennung
 - 4.1.1. Module für „Bildvorbereitung“
 - 4.1.2. Module für Nachbearbeitung der Ergebnisdaten
 - Filtern von erkannten Textdaten
 - Erkennen von Schlüsselwörtern durch unscharfe Suche
 - 4.2. Datenbankstruktur
 - 4.3. Anbindung an bestehenden „ScreenshotManager“
 - 4.3.1. Einbinden der Bibliothek
 - 4.3.2. GUI-Prototyp für Bild-Upload
 - 4.3.3. Verknüpfung der Schlagwortdatenbank mit bestehender Datenbankstruktur
5. Zusammenfassung und Erfahrungen

Zeitplan

| Datum | Meilenstein | Beschreibung |
|------------|----------------|--|
| 31.07.2023 | Expose | Themenfindung, Literaturrecherche, Definition der Forschungsfrage |
| 31.08.2023 | Recherche | Schnittstellendefinition, Datenbankdesign |
| 15.09.2023 | Preprocessing | Vergleich verschiedenster Bildmanipulationsmethoden, anschließendes Sammeln und Auswerten der Daten. Danach Implementierung in Beispielbibliothek |
| 31.09.2023 | Postprocessing | Vergleich und Auswerten der Ergebnisse der verschiedenen Filterstrategien, inklusive unscharfer Suche nach Schlüsselwörtern. Anschließende Implementierung in der Beispielbibliothek |
| 31.11.2023 | Auswertung | Auswertung der extrahierten Textdaten mithilfe der Beispielbibliothek: Qualität der Ergebnisse, Anteil erkannter Worte und Laufzeit der Texterkennung |
| 01.12.2023 | Erster Entwurf | |
| 31.12.2023 | Finale Abgabe | |

Tabelle 1: Zeitplan

Preprocessing (Probekapitel)

Beim sogenannten „Preprocessing“ werden die zu verarbeitenden Bilder gemäß nachfolgender Methoden für die Texterkennung vorbereitet. Ziel der Vorbereitung ist es, Texte im Bild für die Texterkennungs-Engine lesbar zu machen (TODO: QUELLE). Während störende Elemente wie Bildrauschen aus dem Bild entfernt werden sollen, werden Texte ungeachtet der Hinter- bzw. Vordergrundfarbe mit maximalem Kontrast hervorgehoben und von interferierenden „Nicht-Text Elementen“ wie Formen oder graphischen Symbolen isoliert.

2 Stand der Technik

Die im Rahmen dieser Bachelorarbeit verwendete Texterkennungs-Engine Tesseract verfügt in der aktuellen Version (TODO: VERSION) (TODO: QUELLE) über ein neuronales Netzwerk, mit dem auf Basis von sprachspezifischen Trainingsdaten (TODO: QUELLE) Texte in Bildern erkannt werden können.

Laut der Tesseract-Dokumentation gibt es gemäß dem derzeitigen Stand der Technik eine empfohlene Vorgehensweise bei der Vorbereitung von Bildern für die Texterkennung. Jede Methodik bietet dabei wiederum je nach Eigenschaften der Bilddatei verschiedene Vor- und Nachteile:

TODO: [Auflistung einiger Methodiken, mit vor und nachteilen und bildern]

Verwendet man moderne Tesseract-Implementierungen, sind in diesen oft bereits rudimentäre Bildverarbeitungswerkzeuge verfügbar. Mit diesen Werkzeugen werden die eingespeisten Bilder - sofern nicht bereits im richtigen Format - meist automatisch für die Texterkennung vorbereitet.

Diese initiale Bildverarbeitung, das sogenannte „Preprocessing“, folgt in der aktuellen Tesseract-Version folgendem Schema:

TODO: Tesseract-eigenes Preprocessing

(TODO: [Erläuterung der Gründe und Beispiele warum bzw. in welchen Fällen das Preprocessing versagt])

Ohne weitere Einstellungen zu treffen, bewirkt die oben beschriebene Bildverarbeitung zwar ein Umwandeln der Eingangsgrafiken in ein grundsätzlich gutes (TODO: das ist universell aber halt kaka)

2.1 Optimierung

Um die Texterkennung mittels Tesseract weiter zu verbessern, ist es notwendig, das Preprocessing zu spezifizieren indem die Breite an möglichen Eingangsdaten durch Annahmen beschränkt wird.

Im Falle dieser Bachelorarbeit handelt es sich bei den zu verarbeitenden Bildern ausschließlich um digitale Bildschirmaufnahmen von grafischen Benutzeroberflächen. Es kann also angenommen werden, dass sowohl die Perspektive der Aufnahme nicht verzerrt ist und der Kontrast in den meisten Fällen ausreicht, um die relevanten Inhalte zu erkennen. Weiters beinhalten grafische Oberflächen oft farbige Hintergrundflächen zur Trennung von Sektionen, auf die bei etwaigem Thresholding gachtet werden muss.

Um die Ergebnisse der Texterkennung zu optimieren, wurde eine Auswahl an Bildverarbeitungsalgorithmen bestimmt. Anhand einer durch den Menschen verschlagworteten Vergleichsmenge können diese Algorithmen nun mittels der Fehlermetriken A, B und C (TODO: metriken) verglichen werden, um den besten zu ermitteln.

Vergleich

[Erläuterung der verwendeten Methodiken und Erläuterung warum sie sich so gut für Screenshots eignen]

[Auswirkungen der verschiedenen Methodiken auf unterschiedliche Ausgangsbilder]

Postprocessing

Beim „Postprocessing“ werden die durch die Texterkennung ermittelten Biltinhalte gefiltert und aufbereitet. Durch die Filterung können die Ergebnisdaten beispielsweise durch das Entfernen von Füllwörtern auf das für den Nutzer wesentliche reduziert und durch Praktiken wie unscharfes Zuweisen Begriffe etwaige Fehler ausgebessert werden.

3 Stand der Technik

Heutzutage werden vor allem im Gebiet des Natural Language Processing (TODO:) mächtige Werkzeuge zur Analyse von Textbasierten Daten verwendet.

Da im Rahmen dieser Bachelorarbeit ein besonderer Fokus auf dem Verschlagworten der Inhalte liegt, um die Suche innerhalb der Screenshot-Datenbank zu verbessern, sind besonders die Vorgehensweise zur Extraktion der relevanten Daten von großem Nutzen.

Wie auch im Preprocessing gibt es auch im Postprocessing verschiedene Algorithmen, die mit den selben Ausgangsdaten unterschiedlich wertvolle Ergebnisse liefern. Daher ist es ratsam, mehrere Algorithmen zu vergleichen und den für den individuellen Use-case am besten geeigneten zu wählen:

TODO: NLP Folien durchlesen.

Vergleich

TODO: [Auflistung einiger Methodiken, mit vor und nachteilen und bildern]

[Erläuterung der verwendeten Methodiken und Erläuterung warum sie sich so gut für Screenshots eignen]

[Auswirkungen der verschiedenen Methodiken auf unterschiedliche Ausgangsdaten]

Durch den Vergleich ist ersichtlich, dass sich der TODO: in Kombination mit dem TODO: Algorithmus am besten für das Ermitteln von relevanten Schlagworten aus Screenshots von Benutzeroberflächen eignet. Durch die Kombination beschränkt sich die Menge an gefilterten Wörtern ausschließlich auf Füllwörter, die nicht inhaltsspezifisch sind, während durch das Fuzzy Matching kleinere Fehler, verursacht durch die Texterkennung, ausgebessert werden können.

Literatur

4 Optical Character Recognition

Die Bachelorarbeit basiert auf folgender Einstiegsliteratur zum Thema OCR und Tesseract Engine:

- An Overview of the Tesseract OCR Engine [10]
- Advances in Character Recognition [5]
- Optical Character Recognition Systems for Different Languages with Soft Computing [3]
- Character recognition systems : a guide for students and practioners [4]
- Soft computing and signal processing [11]

Auch auf der Website des Tesseract-Projektes beziehungsweise im Wiki des GitHub-Repositories finden sich nebst zahlreichen Publikationen zur Tesseract Engine selbst auch wichtige Information zur richtigen Verwendung von Tesseract:

- Tesseract Documentation [13]

5 Datenbankdesign und UML

Neben dem in der FH Oberösterreich vermittelten Wissen bildet unter anderem folgende Einstiegsliteratur die Basis für das Datenbankdesign und die UML-Diagramme:

- Datenbanksysteme - Eine Einführung [8]
- Datenbanken-Implementierungstechniken [12]
- The unified modeling language user guide [2]
- UML : Database Modeling Workbook [1]
- UML & data modeling : a reconciliation [6]

6 Approximate String Matching

Für das Approximate String Matching, das fuer eine bessere Einteilung des erkannten Texts in bestehende Schlagworte sorgen soll, wird unter Anderem auf folgende Publikationen zurückgegriffen:

- A Guided Tour to Approximate String Matching [9]
- Practical Methods for Approximate String Matching [7]

Quellenverzeichnis

Literatur

- [1] Michael Blaha. *UML : database modeling workbook*. eng. 1. print.. 2013. URL: <https://permalink.obvsg.at/fho/AC11105171> (besucht am 12.06.2023) (siehe S. 10).
- [2] Grady Booch. *The unified modeling language user guide : UML*. eng. 3. print.. Addison-Wesley object technology series. 1999. URL: <https://permalink.obvsg.at/fho/AC08768402> (besucht am 12.06.2023) (siehe S. 10).
- [3] Arindam Chaudhuri. *Optical Character Recognition Systems for Different Languages with Soft Computing*. eng. Studies in Fuzziness and Soft Computing 352. 2017. URL: <https://permalink.obvsg.at/fho/AC12323924> (besucht am 12.06.2023) (siehe S. 10).
- [4] Mohamed Cheriet. *Character recognition systems : a guide for students and practitioners*. eng. 2007. URL: <https://permalink.obvsg.at/fho/AC06408992> (besucht am 12.06.2023) (siehe S. 10).
- [5] Xiaoqing Ding. *Advances in Character Recognition*. eng. IntechOpen, 2012. DOI: 10.5772/2575. (Besucht am 12.06.2023) (siehe S. 10).
- [6] David C Hay. *UML & data modeling : a reconciliation*. eng. 2011. URL: <https://permalink.obvsg.at/fho/YC00337386> (besucht am 12.06.2023) (siehe S. 10).
- [7] Heikki Hyvrö. „Practical Methods for Approximate String Matching“. In: 2003. URL: <https://www.semanticscholar.org/paper/Practical-Methods-for-Approximate-String-Matching-Hyvr%C3%B6/3b2227ae166cbe90b20408da3f2feb75f95afd9c> (besucht am 12.06.2023) (siehe S. 10).
- [8] Alfons Kemper. *Datenbanksysteme : eine Einführung*. ger. 10., aktualisierte u. erw. Aufl.. De-Gruyter-Oldenbourg-Studium. 2015. URL: <https://permalink.obvsg.at/fho/AC12661940> (besucht am 12.06.2023) (siehe S. 10).
- [9] Gonzalo Navarro. „A Guided Tour to Approximate String Matching“. *ACM Computing Surveys* 33 (Apr. 2000). DOI: 10.1145/375360.375365. (Besucht am 12.06.2023) (siehe S. 10).
- [10] Smith R. „An Overview of the Tesseract OCR Engine“. In: 2007. URL: <https://ieeexplore.ieee.org/document/4376991> (besucht am 12.06.2023) (siehe S. 10).

- [11] V. Sivakumar Reddy. *Soft computing and signal processing : : proceedings of 4th ICSCSP 2021*. eng. Advances in Intelligent Systems and Computing ; 2022. URL: https://search-fho.obvsg.at/permalink/f/19351jn/FHO__alma5134174850004527 (besucht am 12.06.2023) (siehe S. 10).
- [12] Gunter Saake. *Datenbanken - Implementierungstechniken : [Architekturprinzipien, Datenstrukturen und Algorithmen, Transaktionsverwaltung und Recovery]*. ger. 3. Aufl.. 2011. URL: <https://permalink.obvsg.at/fho/AC08815950> (besucht am 12.06.2023) (siehe S. 10).
- [13] *Tesseract Documentation*. eng. 23. Mai 2023. URL: <https://tesseract-ocr.github.io/> (besucht am 12.06.2023) (siehe S. 2, 10).