



Fachhochschul-Bachelorstudiengang
SOFTWARE ENGINEERING
A-4232 Hagenberg, Austria

Kategorisierung von lokalisierten Bilddateien mittels OCR

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Science in Engineering

Eingereicht von

Simon Gruber

Begutachtet von Barbara Traxler MSc

Hagenberg, Oktober 2023

Inhaltsverzeichnis

1	Einleitung	1
1.1	Motivation	1
1.2	Herausforderungen	2
1.3	Fragestellung	2
2	Grundlagen	3
2.1	Stand der Technik	3
2.1.1	Texterkennungssysteme	3
2.1.2	Filterung der Ergebnisdaten	3
2.2	Verwendete Technologien	4
2.2.1	Texterkennungssystem	4
2.2.2	Bildbearbeitungswerkzeug	4
3	Konzept	5
3.1	Annahmen	5
3.2	Vergleich	6
3.2.1	Metriken	6
3.2.2	Testaufbau	8
3.3	Verwendete Algorithmen	8
3.3.1	Preprocessing	8
3.3.2	Postprocessing	12
4	Vergleich	14
4.1	Implementierung	14
4.1.1	Vergleichsdaten	14
4.1.2	Programmkomponenten	14
4.1.3	Programmablauf	15
4.2	Analyse	15
5	Zusammenfassung	16
6	Literatur	17
6.1	Optical Character Recognition	17
6.2	Datenbankdesign und UML	17
6.3	Approximate String Matching	18

Inhaltsverzeichnis	iii
Quellenverzeichnis	19
Literatur	19

Kapitel 1

Einleitung

1.1 Motivation

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.

1.2 Herausforderungen

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 erkannter Texte 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-Systems (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.

1.3 Fragestellung

Ziel dieser Bachelorarbeit ist es, herauszufinden, welche Vorgehensweisen bei der Texterkennung zu den besten Ergebnissen führen. Dazu werden die Resultate der Bild- und Textverarbeitung anhand gängiger Fehlermetriken für Texterkennungssysteme verglichen.

Kapitel 2

Grundlagen

2.1 Stand der Technik

2.1.1 Texterkennungssysteme

Optische Texterkennung wird in der Informationstechnik unter anderem dazu verwendet, Texte in verschiedensten Dokumenten oder Grafiken als solche zu erkennen und zu extrahieren. Auf dem Markt gibt es dafür bereits viele kommerzielle Komplettlösungen wie „IronOCR“, „Google Cloud Vision“, „Amazon Textract“ oder „Microsoft Azure Computer Vision“, die oftmals sehr gute Ergebnisse erzielen und sich gut in bestehende Prozesse oder Anwendungen integrieren lassen [15].

Heutige Texterkennungssysteme arbeiten oft mit neuronalen Netzwerken sowie fortgeschrittenen Bildverarbeitungsalgorithmen, um Text in Bilddateien zu erkennen und zu extrahieren. Während es zahlreiche wissenschaftliche Werke zur grundlegenden Funktionsweise von optischen Texterkennungswerkzeugen gibt, wurden die genauen Schritte zur richtigen Vorbereitung der Bilddaten – besonders in Bezug auf Screenshots – nur oberflächlich behandelt.

2.1.2 Filterung der Ergebnisdaten

Das Themengebiet des Natural Language Processing beschäftigt sich mit der Interaktion zwischen menschlicher Sprache und Computern. Natural Language Processing kombiniert Techniken aus der Informatik, Linguistik und dem maschinellen Lernen, um mit menschlicher Sprache umzugehen und beispielsweise Textanalyse, Übersetzungen, Spracherkennung oder Dialogsysteme möglich zu machen. Durch die große Aufmerksamkeit und die häufige Nutzung der Technologien – beispielsweise in digitalen Sprachassistenten – sowie dem Aufkommen von neuronalen Netzwerken wurden in diesem Forschungsgebiet in den letzten Jahren immer wieder Fortschritte erzielt.

Dadurch gibt es zahlreiche wissenschaftliche Ressourcen, die als Grundlage für die in dieser Bachelorarbeit verwendete Vorgehensweise zur Interpretation und Extraktion relevanter Schlagworte aus den erkannten Freitextdaten dienen.

2.2 Verwendete Technologien

2.2.1 Texterkennungssystem

Die Nutzung der in Kapitel 1 erwähnten Anwendungen bzw. Dienstleistungen ist kostenpflichtig und die genaue Vorgehensweise dieser Programme ist nicht öffentlich bekannt.

Aufgrund dieser Tatsachen ist die Wahl des Texterkennungssystems für die prototypische Implementierung dieser Bachelorarbeit auf die seit 2005 unter der Freie-Software-Lizenz „Apache 2.0“ veröffentlichten „Tesseract Open Source OCR Engine“ (kurz: Tesseract) gefallen. Diese basiert seit der Major-Version 4 auf einem neuronalen Netz, durch welches mithilfe von sprachspezifischen Trainingsdaten Texte in Bildern erkannt werden können [13]. Außerdem stellt sie mit mittlerweile über 50.000 Sternen auf der Repository-Hosting-Plattform GitHub eines der beliebtesten Texterkennungssysteme dar [13] [14].

2.2.2 Bildbearbeitungswerkzeug

Als Werkzeug für die Durchführung der notwendigen Bildbearbeitungsschritte wurde die Softwarebibliothek „ImageMagick“ gewählt. Sie ist umfassend dokumentiert, flexibel und lässt sich gut in gängige Programmiersprachen einbinden. Viele in der Bildverarbeitung genutzte Operationen sind außerdem bereits implementiert, was schnelles Prototyping vereinfacht und die Bibliothek zu einer idealen Wahl für die Realisierung von Bildbearbeitungsschritten in der prototypischen Implementierung macht.

Kapitel 3

Konzept

3.1 Annahmen

Um die Texterkennung mittels Tesseract und die anschließende Filterung der Ergebnisdaten zu verbessern, ist es sinnvoll, Anwendungsspezifische Annahmen für den Verarbeitungsablauf festzulegen.

Preprocessing

Eigenschaften von Screenshots

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 die Screenshots keine Transparenz aufweisen, die Perspektive der Aufnahme nicht verzerrt ist und der Kontrast in den meisten Fällen ausreicht, um die relevanten Inhalte zu erkennen. Weiters ist bei der Bildverarbeitung auf farbige Hintergrundflächen zu achten, mit deren Unterstützung Bildelemente in modernen grafischen Oberflächen oft gruppiert oder getrennt werden. Nach Sichtung des zu verarbeitenden Bilddatensatzes fällt zudem auf, dass die manche Screenshots durch das Selektieren mit der Maus sehr eng abgeschnitten wurden. Auch das ist bei der Vorverarbeitung zu berücksichtigen.

Optimieren von Daten für Tesseract

Für die Verwendung von Tesseract ist es wichtig, unabhängig von der Diversität der Ausgangsdaten möglichst einheitliche Bilder zu generieren, die den Trainingsdaten des neuronalen Netzes ähnlich sehen [15]. Während störende Elemente wie Bildrauschen aus dem Bild entfernt werden sollen, sollen Texte unabhängig von der Hinter- bzw. Vordergrundfarbe gut zu erkennen und leicht von Formen oder grafischen Symbolen abzugrenzen sein.

Postprocessing

Filtern von Symbolen

Bei der Texterkennung kommt es manchmal vor, dass grafische Elemente als Unicode-Symbole erkannt werden. Beispielsweise finden sich in den ungefilterten Ergebnisdaten oft Aufzählungszeichen „•“ oder diverse Varianten von Bindestrichen „–“. Diese Zeichen sind gemäß Anwendungsanforderungen nicht relevant für die Schlagwortsuche und können somit entfernt bzw. ignoriert werden.

Mehrsprachigkeit

Eine weitere Anforderung an das Textverarbeitungssystem ist außerdem das Einlesen und Interpretieren mehrsprachiger Bilddateien. So sollen beispielsweise Bilder mit englischen, deutschen oder italienischen Inhalten zugeführt und die Ergebnisdaten richtig verarbeitet werden können. Um eine Filterung für verschiedene Zeichensätze zu ermöglichen und eine Unterstützung für Sprachen mit nicht-lateinischen Schriften zu gewährleisten, werden dynamische Sprachfilter verwendet, die individuell an die jeweilige Sprache angepasst werden können. Um die Ergebnisdaten nicht unnötig aufzubauschen, werden für die initialen Tests und die Beschreibung der generellen Vorgehensweise im Rahmen dieser Bachelorarbeit werden jedoch nur deutsche oder englische Inhalte verarbeitet.

Schlagworte

Für die spätere Suche von Screenshots sollen relevante Schlagworte aus den erkannten Textdaten extrahiert werden. Ein Wort eignet sich dann als Schlagwort, wenn es in relevantem Bezug zum jeweiligen Bild steht und dabei idealerweise eine wichtige Aktion oder Information widerspiegelt. Inhalte, die direkt in der grafischen Benutzeroberfläche ersichtlich sind, eignen sich demnach besonders gut als Suchworte. Damit die Menge an Schlagwörtern allerdings nicht zu unspezifisch wird, sollte es vermieden werden, allgemeine sogenannte Stoppwörter (engl. "Stop words") ohne besondere Semantik, wie „und“, „oder“, in das Ergebnis mit aufzunehmen. Sie beinhalten keine spezifische Information und fördern aufgrund ihrer Häufigkeit das Auftreten von Verwechslungen.

3.2 Vergleich

3.2.1 Metriken

Um die erkannten Ergebnisse unter Verwendung der verschiedenen Pre- und Postprocessing Schritte mittels eines einheitlichen Systems vergleichen zu können, wird auf die in der optischen Texterkennung gängigen Metriken „Character Error Rate“ (CER) und „Word Error Rate“ (WER) zurückgegriffen.

Sowohl die CER als auch die WER sind beliebte Vergleichswerte, die ihren Ursprung in der computergestützten Sprachverarbeitung bzw. automatischen Spracherkennung (ASR) haben. Da OCR und ASR beide darauf abzielen, maschinenlesbaren Text aus nicht-strukturierten Daten zu extrahieren, sind die Prinzipien dieser Metriken auch auf die OCR anwendbar.

3.2.1.1 Word Error Rate

Die Word Error Rate (WER) beschreibt den prozentualen Anteil der falsch erkannten oder fehlenden Wörter eines Textes im Vergleich zu einer Referenz, welche im Falle der folgenden Vergleiche immer alle sichtbaren Texte im Bild repräsentiert. Je niedriger die WER, desto genauer ist der OCR-Vorgang. Um die WER zu berechnen, bildet man die Summe aller notwendigen Ersetzungen, Entfernungen und Einfügungen, um aus dem erkannten Text den Referenztext bilden zu können und teilt sie durch die Anzahl an Wörtern im Referenztext.

Berechnung

Die mathematische Formel für die Word Error Rate lautet somit wie folgt:

$$\text{WER} = \frac{S + D + I}{N}$$

wobei die einzelnen Komponenten folgende Größen darstellen:

- S beschreibt die Anzahl der falsch erkannten Wörter (engl. "Substitutions")
- D beschreibt die Anzahl der im Resultat fehlenden Wörter (engl. "Deletions")
- I beschreibt die Anzahl der im Resultat fälschlicherweise eingefügte Wörter (engl. "Insertions")
- N beschreibt die Gesamtanzahl der Wörter in der Referenz

Vorteile und Nachteile

Die WER spiegelt ohne großen Rechenaufwand direkt wider, wie stark die erkannten Texte der Referenz gleichen. Hierbei werden fehlerhafte Einsetzungen, Löschungen und falsch erkannte Wörter bzw. Teilwörter gleichermaßen gewichtet. Es ist jedoch nicht möglich, die korrekte Reihenfolge der erkannten Wörter darzustellen oder bestimmte wichtige Stellen im Text höher zu gewichten als andere. Auch werden fehlerhaft erkannte Wörter, auch wenn nur ein einzelner Buchstabe falsch ist, als vollwertige Ersetzung wahrgenommen, wodurch die WER selbst bei bis auf wenige Zeichen gut erkannte Texte stark beeinflusst werden kann.

Um also ein umfassendes Bild von der Genauigkeit des Texterkennungssystems zu erhalten, ist es sinnvoll, die Ergebnisse nicht nur anhand der WER, sondern auch noch mindestens anhand einer weiteren Fehlermetrik, wie beispielsweise der CER, zu vergleichen.

3.2.1.2 Character Error Rate

Die Character Error Rate (CER) beschreibt die Anzahl der falsch erkannten oder fehlenden Zeichen im Vergleich zu einem Referenzwort. Je niedriger die CER, desto genauer ist der OCR-Vorgang. Um die CER zu berechnen, bildet man die Summe aller notwendigen Ersetzungen, Entfernungen und Einfügungen, um aus dem erkannten Wort

die Referenz bilden zu können und teilt sie durch die Anzahl an Zeichen im Referenzwort.

Berechnung

Die mathematische Formel für die Word Error Rate lautet somit wie folgt:

$$\text{CER} = \frac{S + D + I}{N}$$

wobei die einzelnen Komponenten folgende Größen darstellen:

- S beschreibt die Anzahl der falsch erkannten Wörter (Substitutions)
- D beschreibt die Anzahl der im Resultat fehlenden Wörter (Deletions)
- I beschreibt die Anzahl der im Resultat fälschlicherweise eingefügte Wörter (Insertions)
- N beschreibt die Gesamtanzahl der Wörter in der Referenz

Vorteile und Nachteile

Die CER fasst in einem Wert zusammen, wie viele Änderungen auf Zeichenebene notwendig sind, um aus dem erkannten Wort das Referenzwort zu bilden. Es ist dabei wie bei der WER nicht relevant, in welcher Reihenfolge diese Zeichen auftreten. Ebenso gibt es keine gesonderte Gewichtung für Ersetzungen, Löschungen oder Einfügungen, wodurch besonders bei kurzen Wörtern auch kleinere Abweichungen bereits zu einer hohen CER führen können.

Durch den detaillierten Vergleich der einzelnen Wörter auf Zeichenebene stellt die CER jedenfalls ein ausreichend gutes Komplement zur WER dar, um in den folgenden Vergleichen genutzt werden zu können.

3.2.2 Testaufbau

Der Testaufbau im Rahmen der Implementierung, beschrieben in Abschnitt 4.1, erlaubt ein dynamisches Verketteten von verschiedenen Bildverarbeitungs- und Textfilterungsschritten. Für einen objektiven Vergleich zwischen den unterschiedlichen Vorgehensweisen und Algorithmen wird eine Grundabfolge der jeweiligen Schritte in einer „Processing-Pipeline“ definiert. Die Ergebnisse können schließlich anhand der in Unterabschnitt 3.2.1 beschriebenen Fehlermetriken mit einer durch den Menschen verschlagworteten Vergleichsmenge abgeglichen werden.

3.3 Verwendete Algorithmen

3.3.1 Preprocessing

Beim sogenannten „Preprocessing“ werden die zu verarbeitenden Bilder für die Texterkennung vorbereitet, um die Qualität der erkannten Textdaten zu verbessern.

Verwendet man moderne Tesseract-Implementierungen, sind in diesen oft bereits rudimentäre Bildverarbeitungswerkzeuge verfügbar [15]. Mit diesen Werkzeugen werden die eingespeisten Bilder – sofern nicht bereits im richtigen Format – automatisch für die Texterkennung vorbereitet. Ohne weitere Einstellungen zu treffen, bewirkt diese Bildverarbeitung zwar ein Umwandeln der Eingangsgrafiken in ein meist gut für Tesseract geeignetes Bild, nichtsdestotrotz ist es jedoch sinnvoll, die Bildverarbeitungsschritte individuell auf die erwarteten Eingangsdaten anzupassen, um sich den in Abschnitt 3.1 definierten optimalen Tesseract-Eingangsdaten anzunähern.

Die folgenden Preprocessing-Schritte basieren auf der empfohlenen Vorgehensweise zur Verbesserung der Output-Qualität laut Tesseract-Dokumentation [13]. Gemäß den obigen Annahmen werden jedoch weder perspektivische Fehler, noch ein eventuelles Rauschen korrigiert. Konkret werden folgende Bildverarbeitungsschritte verglichen:

3.3.1.1 Resampling

Bei Resampling wird die Bildauflösung durch „Neuabtastung“ verändert. Um die für Tesseract optimale [13] Mindestauflösung von 300 dpi zu gewährleisten, muss das Eingangsbild, sofern es die Mindestauflösung unterschreitet, zunächst entsprechend vergrößert werden.

Da Tesseract auf klare und scharfe Kontraste angewiesen ist, um Text korrekt zu identifizieren, eignen sich nicht alle von ImageMagick zur Verfügung gestellten Skalierungsmethoden für die Weiterverarbeitung. Besonders beim Hochskalieren neigen einige Filter dazu, Unschärfen und Artefakte zu erzeugen, die die Genauigkeit der Texterkennung negativ beeinflussen können. [TODO: Beispielbild für Bilineare Skalierung oder nearest-neighbor hier einfügen]. Unter den verschiedenen Resampling-Filtern, die ImageMagick bereitstellt, haben sich insbesondere die Bikubische Interpolation und das Lanczos-Verfahren als für die Texterkennung mit Tesseract geeignet erwiesen [15]:

Bikubische Interpolation

Die Bikubische Interpolation stellt eine Erweiterung der Kubischen Spline Interpolation dar. Für die Berechnung des Ergebniswertes eines Pixels werden bei diesem Verfahren sowohl Pixel aus der ersten, als auch aus der zweiten Nachbarschaft berücksichtigt, wodurch – zu lasten der Laufzeitperformanz – eine hohe Detailtreue erhalten werden kann. Mithilfe von Streuparametern kann beeinflusst werden, wie stark die Übergänge zwischen einzelnen Pixeln geglättet werden bzw. wie scharf die Kanten im berechneten Ergebnisbild sind. Abhängig von der Qualität der Ausgangsdaten können Skalierungsartefakte dadurch weitestgehend vermieden werden.

Bei Verwendung der Bikubischen Interpolation als Resampling-Algorithmus fällt auf, dass Bilder und Grafiken durch die Glättung für das menschliche Auge ansprechender wirken. Tesseract jedoch profitiert stark von klaren Texten und hohen Kontrasten, weswegen diese Art des Resamplings keine ideale Basis für die nächsten Preprocessing-schritte bildet.

Lanczos Filterung

Das Lanczos-Verfahren erlaubt es, Bilder beim Resampling präzise zu rekonstruieren. Es verwendet eine Fensterfunktion, basierend auf dem nichtnormierten Sinus Cardinalis, auch bekannt als Samplingfunktion $\text{sinc}(x)$ und erzeugt im Gegensatz zur Bikubischen Interpolation weniger stark geglättete Ergebnisse, dafür sind diese jedoch meist schärfer. Durch die Verwendung der aufwändigen Samplingfunktion ist Resampling nach der Lanczos-Methode vergleichsweise rechenintensiv.

Aufgrund der höheren Bildschärfe und den dadurch deutlich klareren Texten fällt die Wahl des Skalierungsverfahrens für die weiteren Schritte auf die Lanczos-Methode.

3.3.1.2 Rahmen

Befindet sich Text zu nah am Rand des Bildes, kommt es vor, dass dieser nicht richtig erkannt werden kann. Ebenso kann auch ein zu großer einfärbiger Rahmen am Rand des Bildes dazu führen, dass Bildsektionen fälschlicherweise als „leer“ erkannt und übersprungen werden, wodurch der zu erkennende Text nicht in die Ergebnisdaten mit aufgenommen wird.

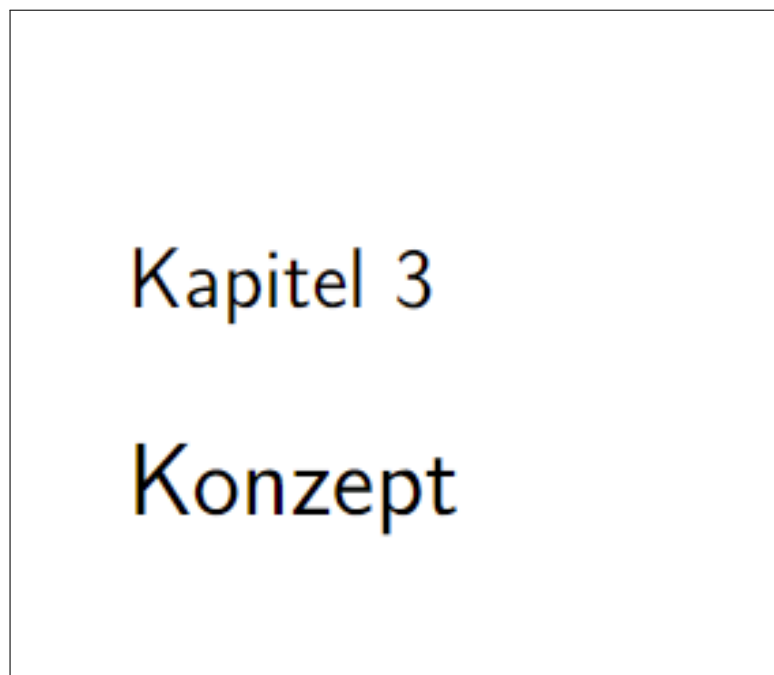


Abbildung 3.1: Ein im Verhältnis zur Bildgröße zu großer einfärbiger Rahmen

3.3.1.3 Binarisierung

Das Erzeugen eines Binärbildes ist durch Anwendung von Segmentierungsverfahren möglich. Schwellenwertverfahren (engl. "Thresholding") bilden eine Untergruppe der Segmentierungsverfahren und werden genutzt, um Graustufenbilder Pixel für Pixel in binarisierte Ergebnisbilder mit zwei Segmenten, also einem Vordergrund und einem Hintergrund umzuwandeln. Der dazu notwendige Schwellenwert kann entweder fix definiert

oder anhand von verschiedensten Algorithmen ermittelt werden. Ziel ist es, durch die Binarisierung textuelle Bildinhalte unabhängig von der eigentlichen Vorder- und Hintergrundfarbe mit ausreichendem Kontrast darzustellen. Somit ist das Texterkennungssystem in der Lage, die einzelnen Textelemente und deren Inhalte besser zu identifizieren und zu verarbeiten.

ImageMagick bietet eine Vielzahl an Thresholding-Algorithmen, deren Eignung in Abschnitt 3.2 verglichen wird.

Feste Schwellenwertmethode

Ein häufig für die Bildsegmentierung genutztes Verfahren ist die feste Schwellenwertmethode, auf Englisch auch „Fixed Thresholding“ genannt. Bei diesem Bildverarbeitungsverfahren wird eine vordefinierter Grenzwert auf einzelne Pixelwerte angewandt. Liegt der Pixelwert über dem festgelegten Schwellenwert, gilt er als Teil des Vordergrunds, andernfalls als Hintergrund. Somit können Objekte, also die einzelnen Buchstaben in den Grafikdateien, von ihrem Hintergrund getrennt werden.

Obwohl das fixe Thresholding durch seine Simplizität einen relativ geringen Berechnungsaufwand benötigt und daher eine hohe Performance aufweist, sind die Ergebnisdaten abhängig von dem Schwellenwert und dem Ausgangsbild meist unzufriedenstellend. So ergibt es sich oft, dass die eigentlich bunten grafischen Elemente der Benutzeroberfläche aufgrund ihrer Helligkeit über dem Schwellenwert liegen. Dadurch werden sie, genau wie der Text, als Vordergrund wahrgenommen und die gesamte Fläche wird einfärbig. Dadurch kann jeglicher Text innerhalb dieser Fläche nicht vom Texterkennungssystem erkannt werden.

Adaptive Schwellenwertmethode

Bei der adaptiven Schwellenwertmethode wird der Schwellenwert auf Basis der lokalen Eigenschaften eines Bildbereichs angepasst, der durch die sogenannte „Blockgröße“ definiert wird. Innerhalb eines Blocks wird schließlich ein fixer Schwellenwert ermittelt. Dadurch können im Gegensatz zur festen Schwellenwertmethode verschiedenfarbige Texte auf Hintergründen unterschiedlicher Helligkeit besser abgegrenzt werden und die Menge an erkanntem Text wird erhöht.

Dreiecks-Schwellenwertmethode

Das Dreiecks-Schwellenwertverfahren verwendet das Histogramm eines Bildes, um einen globalen Schwellenwert zu ermitteln. Innerhalb des Histogramms wird eine Linie vom Höchstwert (engl. "Peak") zum Minimum gezeichnet und ermittelt die Normale mit der maximalen Länge. Dieses Verfahren erzielt die besten Ergebnisse, wenn die zu extrahierenden Elemente Intensitätswerte aufweisen, die an der Basis des ermittelten Peaks liegen. Für Screenshots von UI-Elementen mit komplexer Struktur und farblich stark variierenden Komponenten ist es eher nicht geeignet.

Schwellenwertmethode nach Otsu

Bei dem Schwellenwertverfahren nach Otsu wird der globale Schwellenwert für ein Bild

anhand des jeweiligen Histogramms ermittelt. Aufgrund dieser Eigenschaften funktioniert das Verfahren am besten, wenn das Histogramm des Bildes eine bimodale Verteilung aufweist, also zwei klare Spitzen hat. Enthält ein Bild jedoch starkes Hintergrundrauschen, funktioniert die Schwellenwertermittlung nur unzuverlässig. Weist es lokale Helligkeitsunterschiede auf, wie es bei grafischen Oberflächen mit ihren unterschiedlich eingefärbten Oberflächensektionen oft der Fall ist, entstehen Dank der Bestimmung eines einzelnen globalen Wertes für das gesamte Bild ähnliche Probleme wie bei der fixen Schwellenwertmethode.

Schwellenwertmethode nach Kapur

Die Schwellenwertmethode nach Kapur, Sahoo und Wong zielt darauf ab, einen Schwellenwert zu finden, der die Entropie zwischen den Vorder- und Hintergrundregionen maximiert.

Dieses Schwellenwertverfahren liefert gute Ergebnisse bei Bildern mit starker Varianz der Vorder- und Hintergrundkontraste bzw. eine breite Helligkeitsverteilung aufweist.

3.3.2 Postprocessing

Da die verarbeiteten Bilddaten bzw. deren extrahierte Textdaten später durch eine Schlagwort-basierte Suchfunktion durch den Nutzer auffindbar sein müssen, müssen die Ergebnisdaten im Rahmen des Postprocessings weiterverarbeitet werden. Ziel ist es, die Redundanz innerhalb des Datensets zu reduzieren. Ebenso sollen falsch erkannte Ergebnisdaten identifiziert und aus der Schlagwortmenge entfernt werden.

3.3.2.1 Normalisierung

Um die aus der Texterkennung gewonnenen Daten zunächst für die weitere Filterung vorzubereiten, ist es sinnvoll, die Redundanz der Daten möglichst zu reduzieren und die einzelnen Wörter zu normalisieren bzw. zu standardisieren. Beispielsweise kann durch das Umwandeln aller Textdaten in Kleinbuchstaben die Variation der Daten eingeschränkt werden, ohne jedoch für die Suche relevante Information zu verlieren.

3.3.2.2 Vermeidung von Duplikaten

Nach der Normalisierung werden Duplikate innerhalb der erkannten Textdaten entfernt. Dies verringert ebenfalls die Redundanz der Textdaten und steigert die Effizienz der nachfolgenden Filterverfahren.

3.3.2.3 Filterung anhand der Genauigkeit

Tesseract stellt im Rahmen der Texterkennung auch immer Metadaten zu den erkannten Texten zur Verfügung. Bei erkannten Wörtern wird beispielsweise immer eine Genauigkeit (engl. "Confidence") mit angegeben. Sie bestimmt, mit welcher Sicherheit ein Texterkennungssystem das jeweilige Wort erkannt hat, wobei Wörter mit hoher Confidence eher richtig, mit niedriger Confidence eher falsch erkannt wurden.

Der Confidence-Filter prüft die jeweiligen Wörter auf ihre Metadaten und verwirft Wörter, deren Confidence unter einem fixen Schwellenwert liegt. Je nach Einstellung bzw. „Härte“ des Filters wird die Anzahl der falsch erkannten Inhalte innerhalb der Schlagwortmenge drastisch reduziert. Ist der Filter zu streng eingestellt, werden jedoch insgesamt weniger Worte in die Ergebnisse mit aufgenommen und es kann vorkommen, dass auch ursprünglich korrekt erkannte Wörter aufgrund eines niedrigen Confidence-Wertes verworfen werden.

3.3.2.4 Filterung anhand der Wortlänge

Verarbeitet das Texterkennungssystem Texte mit unregelmäßigen Abständen oder grafischen Artefakten in der Schrift, werden statt des eigentlichen Wortes fälschlicherweise oft relativ kurze Symbolkombinationen erkannt. Um diese Kombinationen aus den Ergebnisdaten zu entfernen, können Zeichenketten mithilfe des Wortlängenfilters ungeachtet ihres Inhaltes verworfen werden.

Zusätzlich kann dieser Filter an die Anforderung des Zielsystems angepasst werden. So sind beispielsweise Wörter, die weniger als zwei Zeichen beinhalten, für die Schlagwortsuche grundsätzlich nicht relevant.

3.3.2.5 Sprachabhängige Filterung mittels Regular Expressions

Nachdem die zu filternden Textdaten durch vorherige Schritte weitestgehend vorverarbeitet wurden, werden die Ergebnisdaten ein letztes Mal mithilfe von regulären Ausdrücken (engl. "Regular Expressions") durchsucht. Aufgrund der guten Erweiterbarkeit der Regular Expressions ist es einfach, für jede Sprache einen individuellen Filter anzulegen, der den jeweiligen Zeichensatz beschriftet und unbekannte Sonderzeichen oder Symbole entfernt. So sind beispielsweise im Deutschen Umlaute erlaubt, während häufig auftretende, jedoch unerwünschte Symbole wie das phonetische Zeichen „æ“ explizit entfernt werden können.

Kapitel 4

Vergleich

4.1 Implementierung

4.1.1 Vergleichsdaten

Erklärung und beispielhaftes Vorzeigen der Ausgangsdaten bzw. der durch den Menschen verschlagworteten Vergleichsdaten.

4.1.2 Programmkomponenten

4.1.2.1 Bibliotheken

Auflistung und kurze Erklärung der in dem Programm verwendeten Bibliotheken in Anlehnung an die in Abschnitt 2.2 aufgezählten Bibliotheken, nur wird hier eben das KONKRETE C#NuGet besprochen

4.1.2.2 Verarbeitungssystem

Erklärung des dynamischen Processing-Frameworks

4.1.2.3 Verarbeitungskette

Detaillierte Erklärung der für diese BA verwendeten Verarbeitungskette bzw. deren Ablauf inklusive Code-Auszügen.

4.1.3 Programmablauf

4.1.3.1 Mehrfachverarbeitung

4.1.3.2 Vergleich mit Soll-Daten

4.1.3.3 Automatische Berichterstellung

4.2 Analyse

Zeigen des erstellten Reports

Ausarbeiten des erstellten Reports

Kapitel 5

Zusammenfassung

Hier werden die Ergebnisse der Bachelorarbeit zusammengefasst und die Ergebnisse besprochen.

Kapitel 6

Literatur

6.1 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]

6.2 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]

6.3 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. 17).
- [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. 17).
- [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. 17).
- [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. 17).
- [5] Xiaoqing Ding. *Advances in Character Recognition*. eng. IntechOpen, 2012. DOI: 10.5772/2575. (Besucht am 12.06.2023) (siehe S. 17).
- [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. 17).
- [7] Heikki Hyvärö. „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. 18).
- [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. 17).
- [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. 18).
- [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. 17).

- [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. 17).
- [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. 17).
- [13] *Tesseract Documentation*. eng. 23. Mai 2023. URL: <https://tesseract-ocr.github.io/> (besucht am 12.06.2023) (siehe S. 4, 9, 17).
- [14] *Tesseract Repository*. eng. 23. Mai 2023. URL: <https://github.com/tesseract-ocr/tesseract> (besucht am 04.01.2024) (siehe S. 4).
- [15] *TODO: MISSING SOURCE*. eng. 23. Mai 2023. URL: <https://example.com/todo> (besucht am 04.01.2024) (siehe S. 3, 5, 9).