Semiautomatische Metadaten-Extraktion und Qualitätsmanagement in Workflow-Systemen zur Digitalisierung historischer Dokumente

Dissertation zur Erlangung des
naturwissenschaftlichen Doktorgrades
der Julius-Maximilians-Universität Würzburg

vorgelegt von

Hendrik Schöneberg

Würzburg 2014

Eingereicht am 22. April 2014
bei der Fakultät für Mathematik und Informatik

1. Gutachter: Prof. Dr. Jürgen Albert
2. Gutachter: Prof. Dr. Frank Puppe
Tag der mündlichen Prüfung: 06. Oktober 2014

Danksagung

Meinen herzlichen Dank möchte ich Herrn Prof. Dr. Jürgen Albert aussprechen für die Idee zu dieser Arbeit, die zahlreichen Anregungen, wertvollen Hinweise und Diskussionen, ohne die ein Gelingen dieser Arbeit nicht möglich gewesen wäre.

Auch Herrn Prof. Dr. Frank Puppe danke ich für seine Anregungen in Diskussionen und sein Interesse an dieser Arbeit.

Besonders möchte ich auch Herrn Dr. Hans-Günter Schmidt danken für die jederzeit spannende Zusammenarbeit in Projekten der Universitätsbibliothek und für sein stets offenes Ohr und guten Rat.

Ich danke auch meinen Arbeitskollegen für das kollegiale und kreative Arbeitsumfeld, sowie den studentischen Hilfskräften, Diplomanden, Bachelorstudenten und Projektpraktikanten, die zur Entstehung dieser Arbeit beigetragen haben.

Natürlich danke ich auch meiner Familie für die Unterstützung bei der Korrektur dieser Arbeit und für die Grundsteine, die sie für meinen Weg gelegt haben.

Inhaltsverzeichnis

1 Einführung

Die Digitalisierung von Dokumenten ist ein Bestreben, welches in den letzten Jahren fortwährend an Bedeutung gewonnen hat. Insbesondere für wertvolle historische Dokumente bietet ein Digitalisat viele Vorteile: In ihrer digitalen Form können Dokumente einem breiten Publikum zugänglich gemacht werden, ohne dabei das empfindliche Original zu gefährden. Weiterhin können sie so in mehreren redundanten Kopien für die Nachwelt bewahrt werden, vielfältigen Verfahren der Information Extraction unterzogen, flexibler präsentiert und für leichtere Zusammenarbeit mit den Digital Humanities präpariert werden.
Häufig wird der Begriff Digitalisierung auch dann verwendet, wenn lediglich die reinen Bilddaten einer Quelle erfasst werden. Die Semantik des Werks, sein logischer und physikalischer Aufbau, Informationen zum Autor, die Enstehungsgeschichte, kurzum die Metadaten des Werks werden oft vernachlässigt. Gerade für wissenschaftliches Arbeiten auf einer Quelle sind diese Metadaten aber essenziell, beispielsweise um innerhalb eines Werks nach Begriffen suchen oder um Querverweise in weiterführende Literatur herstellen zu können. Metadaten bieten dem Betrachter weiterhin deutliche Mehrwerte in der Präsentation: Sie erlauben intuitiveres Navigieren innerhalb eines Digitalisats, beispielsweise durch Sprungverweise zu bestimmten Kapiteln und Passagen oder helfen beim Verständnis durch synoptische Visualisierungen wie Überlagerung von Original und Übersetzung.
Die Bereitstellung dieser Metadaten ist jedoch ein zeitaufwändiger und fehleranfälliger Prozess, der häufig nur von menschlichen Experten vorgenommen werden kann. Wünschenswert wären automatisierte Verfahren zur Metadaten-Extraktion, die in den Digitalisierungs-Prozess mit eingebunden werden können. Aus dem Bereich des Information Retrievals und der Named Entity Recognition existieren dazu viele erprobte Verfahren für neusprachliche Dokumente.

Diese Arbeit spezialisiert sich auf die Domäne der historischen Dokumente, welche einige zusätzliche Schwierigkeiten mit sich bringt: Einerseits ist die Rechtschreibung in historischen Dokumenten häufig nur konventionell vereinbart und unterliegt nicht der strikten Reglementierung zeitgenössischer Werke. Weiterhin sorgt ein schlechter Erhaltungszustand oder Beschädigung der Dokumente dafür, dass Passagen heuristisch rekonstruiert werden müssen. Digitale Transkripte dieser Dokumente besitzen einen hohen Fehleranteil, weil sie entweder aus OCR-Verfahren gewonnen wurden, die mit dem schwierigen Schriftbild häufig Probleme haben, oder von menschlichen Übersetzern bereitgestellt wurden. Die daraus resultierende hohe Schreibweisenvarianz macht es nachgeschalteten automatisierten Verfahren schwer, Informationen zu extrahieren oder Klassifikationen vorzunehmen.
In dieser Arbeit sollen Verfahren vorgestellt werden, die mit der hohen Fehlerquote und Schreibweisenvarianz in dieser Domäne umgehen und eine semiautomatisierte Metadaten-Extraktion vornehmen können. Dabei wird der syntaktisch häufig konsistentere Kontext eines Terms statistisch analysiert, auf Muster untersucht und die Anwendbarkeit dieser Informationen zur Klassifikation unbekannter Terme bewertet. Die Extraktion ist insofern semiautomatisch, als vom Anwender dabei in Abhängigkeit vom gewählten Verfahren Gutmuster oder Feedback verlangt werden können. Insbesondere werden ein auf Conditional Random Fields gestütztes Verfahren, ein auf dem Vector Space Model aufbauendes Verfahren sowie ein Verfahren aus dem Bereich Machine Learning evaluiert.
Es soll gezeigt werden, dass auf diese Weise auch in syntaktisch schwierigen Domänen Metadaten gewonnen werden können, wodurch die Notwendigkeit menschlicher Beteiligung reduziert wird. Weiterhin wird illustriert, dass die so gewonnenen Metadaten im Rahmen eines Workflow-Systems zur Digitalisierung historischer Dokumente eine frühzeitige Fehlererkennung ermöglichen und die Durchlaufzeit von Digitalisierungsprojekten dadurch erheblich verringert werden kann. Schließlich wird am Beispiel einer Präsentations-Plattform im Internet gezeigt, wie die Metadaten dem Nutzer eine originalgetreuere und umfassendere Darstellung des Originals bieten.
Diese Arbeit zeigt weiterhin, dass ein Workflow-System, welches sich automatisiert gewonnene Metadaten zu Nutze macht, sei es zur Fehlererkennung oder zur Visualisierung unterschiedlicher Aspekte eines Digitalisats, eine Innovation darstellt, die

sich von anderen Workflow-Systemen deutlich abgrenzt. Dazu wird das Workflow-System der Digitalisierungsabteilung der Universitätsbibliothek Würzburg vorgestellt und ein Vergleich zu anderen im Einsatz befindlichen Workflow-Systemen gezogen.

1.1 Motivation

Digitalisierungsprojekte haben in den letzten Jahren fortwährend an Bedeutung gewonnen. Die Digitalisierung von Dokumenten avanciert dadurch mehr und mehr zum alltäglichen Handwerk für Bibliotheken, Archive und Museen. Das Bestreben, digitalisierte Werke einem großen Publikum zur Verfügung zu stellen, hat seit 2004 nicht zuletzt durch die Beteiligung von Initiativen wie Google Books[1] noch verstärkten Aufschwung erfahren und dazu geführt, dass der Aufbau einer Infrastruktur für Wissenschaft, Forschung und Gesellschaft zu einer globalen Zielsetzung geworden ist.

Die Vorzüge der Digitalisierung sind dabei vielfältig: Ein digitalisiertes Objekt kann einerseits über das Internet einem breiten Publikum zugänglich gemacht werden, ohne sich um Öffnungszeiten, Ausstellungsräume oder Sicherheit Gedanken machen zu müssen. Auf der anderen Seite kann durch die Verwendung einer digitalen Kopie vermieden werden, das Original durch die Benutzung zu beschädigen. Im Gegensatz zum Original ist es bei digitalisierten Objekten darüber hinaus jederzeit möglich, eine identische Kopie anzulegen und an verschiedenen Orten redundant zu speichern, wodurch man den langfristigen Erhalt des Objekts gewährleisten kann. Nicht zuletzt eröffnet die digitale Form die Möglichkeit zur Zusammenarbeit mit den Digital Humanities und erlaubt es, Verfahren aus dem Bereich der Information Extraction anzuwenden.

In dieser Arbeit wird der Begriff der Digitalisierung synonym mit der Retrodigitalisierung, also der Digitalisierung teilweise sehr alter Handschriften, Landkarten

[1] http://books.google.de/

und Drucke verwendet. Das Digitalisierungszentrum der Universität Würzburg hat sich auf diese Teildisziplin spezialisiert.

Bei der (Retro-)Digitalisierung werden üblicherweise zunächst reine Bilddaten erfasst. Die Abbildung ist allerdings nur ein Aspekt des Originals: Der physikalische und logische Aufbau eines Werkes sowie dessen Inhalt oder Semantik werden dadurch nicht erfasst. Für eine möglichst detailgetreue Digitalisierung dürfen diese und weitere Aspekte des originalen Objekts jedoch nicht vernachlässigt werden. Beispielsweise ist es für ein gründliches wissenschaftliches Arbeiten auf einem Dokument unerlässlich, dass man bestimmte Textpassagen durchsuchen und aus Suchmaschinen darauf verweisen kann. Dazu wird ein Transkript oder ein Volltext des Originals benötigt. Weiterhin sind Querverweise in verwandte oder weiterführende Dokumente von großem Interesse; auch für diese Korrelation benötigt man Hintergrundwissen. Bibliographische Metadaten, Informationen zur Herstellung, Interpretationen oder Übersetzungen sind weitere wichtige Aspekte eines Werks, die bei einer Digitalisierung nicht vernachlässigt werden sollten.


Von S. Kilian den Ersten Bischoff zü
Wirtzburg

Wie S. Kilian gein Wirtzbürg Komen
vnd den Christen Glauben da gepredigt hatt

ZU DISER Zeit ware Kunig in Francken Dieterich in der Zale der Dreyzehendt. Dem waren vnderthan *Galler Land, Niderland, Hessen, Francken, Düringen, Schwaben, Bayrn, Burgund, Lotaringen* vnd der gantz *Reinstrom* mit Jren Hertzogen vnd Grauen. Vnd wiewol die *Francken jenseits Reins* den Christen glauben vorlengest angenomen hetten, So pliben doch die Francken, Bayrn, hessen, düringen, Sachssen vnd was diser Seit Reins Ligt, Jn Jrem Haidischem glauben. Als nun solches *S. Chilian* angeZaigt warde, Name er aus seiner Schuelversamlung Zwen gefertten Zu Jme, Nemlich *Colman* vnd *Dietman* genant; mit den Schiffet er Jn der Gallen Landt, Jtzundt Franckhreich gehaiss(en), vnd Zoge further gen Rome, Von Rome in Teutsche Lande hieher gen Wirtzburgk, Fieng an daselbst vnd herumber den Heyligen Christen glauben Zu Predigen, dann er Als oblaut von seinen Elttern ein geborner Sachß

Abbildung 1.1: Fries Chronik, 16. Jahrhundert, Transkript mit beispielhaften orthographischen Besonderheiten.

Die Erstellung eines polymorphen Digitalisats, also eines Digitalisats, welches möglichst viele Aspekte des Orginaldokuments zu erfassen versucht, ist häufig ein zeitintensiver, manueller Prozess. Insbesondere bei der Retrodigitalisierung erfordert die Erstellung werkbegleitender Volltexte das manuelle Eingreifen eines Experten:

Graphem	Schreibweisenvarianten
<a>	<á, å, ah, aa, ai, ae, a̋>
<e>	<eh, ee, ei, ey, a̋, e̋ , ä>
<i>	<j, y, ÿ, ie, iee, i̋, ij, ye, ih, jh, ieh, yh>
<o>	<oh, ó, oe, oi, oy, oo>
<u>	<ú, ű, ů, v, w, uh, wh, ůh, uy>
<ä>	<a̋, e, a, æ, ae, äh>
<ü>	<ű, u, ů, v, ü, ÿ, y, w, ue, üe, üh, uy>
<ö>	<ő, ó, o, öh, oe, öe, oe̋, œ>

Tabelle 1.1: Schreibweisenvarianten in hist. Dokumenten (Auszug).

Alte Werke zeichnen sich häufig durch eine sehr spezifische Domäne und syntaktische Besonderheiten aus, die automatisierte Vorgänge oft fehlschlagen lassen. In [Hauser u. a. (2007)] wird die hohe Schreibweisenvarianz historischer Dokumente beschrieben. Tabelle 1.1 zeigt beispielhaft Schreibweisenvarianten für einige Grapheme.

Abbildung 1.1 zeigt ein gerendertes Transkript der „Fries Chronik"[2], einer Chronik aus dem 16. Jahrhundert in Frühneuhochdeutsch. In der Abbildung sind drei unterschiedliche Schreibweisen des Terms „Würzburg“ zu erkennen, obwohl sie nur wenige Zeilen voneinander trennen.

Schreibweisenvarianten für 'Würzburg' (Auszug)		
Herbipolis	Wirceburgum	Wirciburc
Wirtzburg	Wirzburg	Wirziaburg
Würtzburg	Würtzb	wurtzb
Wurtzburg	wurtzburg	würtzburgk
würtzburg	W	würtzberg
Wurtzb	Würzburg	wurzburg
		...

Tabelle 1.2: Schreibweisenvarianten für 'Würzburg' (Auszug).

Tabelle 1.1 zeigt einige beispielhafte Schreibweisenvarianten für den Ortsnamen „Würzburg“. Für automatisierte Verfahren stellen diese und weitere syntaktische Besonderheiten ein nicht-triviales Problem dar und machen das Eingreifen mensch-

[2] http://franconica.uni-wuerzburg.de/ub/fries/index.html

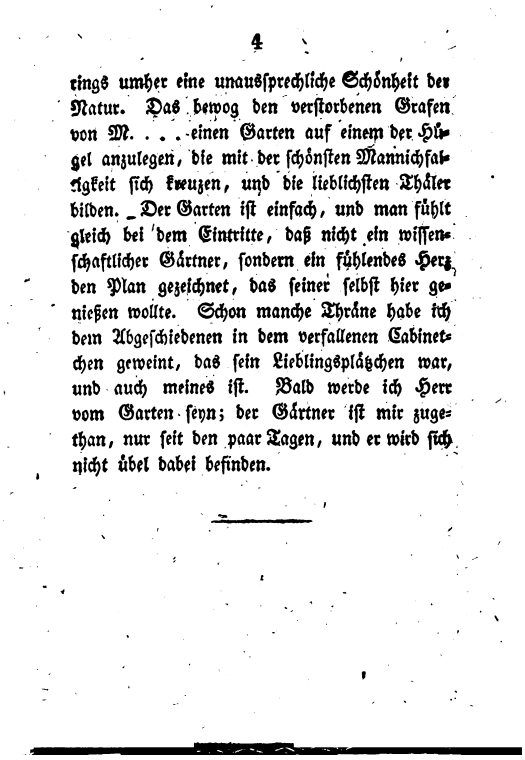
licher Experten notwendig. Kapitel 2 stellt die Probleme im Zusammenhang mit der automatischen Extraktion von Metadaten in der Domäne der Retro-Digitalisierung genauer vor.

In dieser Arbeit werden Verfahren vorgestellt, die die syntaktischen Schwierigkeiten der Domäne zu kompensieren vermögen: Kapitel 5 beschreibt zunächst einen Ansatz, der auf Conditional Random Fields basiert, vergleicht diesen dann mit einem auf einer Verallgemeinerung des Vector Space Models beruhenden Methode und schließt mit der Vorstellung eines auf Methoden des Machine Learning beruhenden Verfahrens. Besonderer Wert soll auf die Eignung dieser Verfahren zum Einsatz in einer Verarbeitungskette in einem Workflow-System zur Digitalisierung historischer Dokumente gelegt werden, da anhand der gewonnenen Metadaten Fehler in der Produktionskette gefunden werden können und dem Nutzer eine umfassendere Präsentation des Quelldokuments ermöglicht wird.

Unlängst haben sich auch Größen wie Google in Initiativen wie *Google Books*[3] der massenhaften Digitalisierung verschrieben. Üblicherweise wird dabei ein Werk eingescannt und in beschränktem Umfang weltweit zugänglich gemacht [*Heise online* (2013)]. Die gescannten Seiten werden einer automatisierten Text-Erkennung und Indizierung unterzogen, so dass ihre Inhalte von Suchmaschinen künftig gefunden werden können. Der Qualität dieser optischen und inhaltlichen Erschließung sind dabei aber deutliche Grenzen gesetzt: Nicht nur weicht das digitale Abbild häufig stark vom Original ab, auch werden Fehler in der automatisierten Text-Erkennung nicht ausgemerzt, bei Nicht-Gelingen gibt es sogar keinerlei Volltexte (siehe [*Le-Publikateur* (2013) und *Frankfurter Rundschau* (2013)].

Abbildung 1.2 illustriert beispielsweise eine Seite aus *Die Leiden des jungen Werther*[4] von Johann Wolfgang von Goethe, welches von Google digitalisiert der Öffentlichkeit zur Verfügung gestellt wurde. Zu beachten sind insbesondere die grobe Qualität des Digitalisats und das Fehlen jeglicher werksspezifischer Navigations-

[3] http://books.google.com/
[4] http://books.google.de/books?id=bHwHAAAAQAAJ&printsec=frontcover



4

rings umher eine unaussprechliche Schönheit der Natur. Das bewog den verstorbenen Grafen von M. . . . einen Garten auf einem der Hügel anzulegen, die mit der schönsten Mannichfaltigkeit sich kreuzen, und die lieblichsten Thäler bilden. Der Garten ist einfach, und man fühlt gleich bei dem Eintritte, daß nicht ein wissenschaftlicher Gärtner, sondern ein fühlendes Herz den Plan gezeichnet, das seiner selbst hier genießen wollte. Schon manche Thräne habe ich dem Abgeschiedenen in dem verfallenen Cabinetchen geweint, das sein Lieblingsplätzchen war, und auch meines ist. Bald werde ich Herr vom Garten seyn; der Gärtner ist mir zugethan, nur seit den paar Tagen, und er wird sich nicht übel dabei befinden.

Abbildung 1.2: Die Leiden des jungen Werther, Johann Wolfgang von Goethe. Digitalisiert von Google Books.

elemente wie Sprungverweise in bestimmte Kapitel oder Abschnitte.

Google Books ist nur ein Beispiel einer Digitalisierungs-Initiative, welche den Schwerpunkt auf die Bilddaten legt und nur wenig Aufwand betreibt, werksspezifische Metadaten zu erfassen. Insgesamt bieten nur wenige am Markt befindliche Workflow-Systeme überhaupt die Möglichkeit, Digitalisate manuell mit Metadaten anzureichern. Verfahren zur automatischen Erfassung fehlen völlig. Abschnitt 9.3 geht detaillierter auf unterschiedliche Workflow-Systeme ein.

1.1.1 Workflow-System der Digitalisierungsabteilung der Universitätsbibliothek Würzburg.

Die Digitalisierungsabteilung der Universitätsbibliothek Würzburg hat sich auf die Digitalisierung von Dokumenten wie historischen Landkarten, wertvollen mittelalterlichen Manuskripten und Grafiken spezialisiert. Aktuell (Stand 2013) befasst sie sich mit der Digitalisierung prestigeträchtiger Handschriften der Würzburger Dombibliothek [Würzburg (2014a)], welche auf einem eigens geschaffenen Präsentations-Portal betrachtet und erforscht werden können. Dieses Portal, Virtuelle Bibliothek Würzburg[5] [Würzburg (2014b)] getauft, vereint dazu Metadaten und Aufnahmen der Dokumente in einer Vielzahl synoptischer Visualisierungen, welche es sich zum Ziel machen, dem Betrachter viele, über die reinen Bilddaten hinausgehende Eindrücke des Originals zu vermitteln.

In Kooperation mit dem Digitalisierungszentrum der Universität Würzburg wurde dazu ein Workflow-System entwickelt, welches sich im Kontrast zu den auf Masse ausgelegten Verfahren der originalgetreuen Digitalisierung und hochwertigen inhaltlichen Erschließung eines Dokuments widmet. Die dabei verwendeten Algorithmen zur Metadaten-Extraktion und zur Qualitätssicherung müssen mit den Besonderheiten der Domäne und der Digitalisate umgehen können. Die gewonnenen Metadaten lassen sich neben den inhaltlichen Mehrwerten zusätzlich dazu verwenden, Produktionsfehler automatisiert zu erkennen, wodurch nicht nur die Durchlaufzeit deutlich reduziert, sondern auch die Digitalisierungsqualität maximiert wird. Kapitel 9 stellt das Workflow-System und die Einbindung der automatisierten Verfahren zur Metadaten-Extraktion vor und evaluiert die Leistungsfähigkeit. In Abschnitt 9.3 wird ein Vergleich des Systems zu anderen, derzeit im Einsatz befindlichen Lösungen gezogen.

[5] http://vb.uni-wuerzburg.de/

1.2 Gliederung der Arbeit

Das vorangegangene Kapitel war eine kurze Einführung in die Thematik und hat die Problemstellung sowie die praxisorientierte Umsetzung sowie die vielfältigen Verwertungsmöglichkeiten der vorgestellten Lösungen angesprochen. Nachfolgend soll nun ein Überblick über die kommenden Kapitel gegeben werden.

Kapitel 2 erläutert die Absicht dieser Arbeit, Informationen aus dem Kontext zur Klassifikation verwenden zu wollen, genauer und stellt zentrale Definitionen für häufig verwendete Begriffe der nachfolgenden Kapitel zur Verfügung.

In *Kapitel 3* werden Arbeiten vorgestellt, die sich mit der Auswertung kontextueller Informationen beschäftigen und ihre Anwendbarkeit auf unsere Zielsetzung diskutiert.

Kapitel 4 stellt prominente Verfahren aus dem Bereich der Named Entity Recognition sowie des Information Retrievals vor und erläutert die Abgrenzungskriterien zu den vorgestellten Verfahren dieser Arbeit.

Kapitel 5 demonstriert schließlich drei Ansätze, die anhand kontextueller Informationen unbekannte Terme klassifizieren können und evaluiert ihre Leistungsfähigkeit auf historischen und syntaktisch schwierigen Texten. Über eine Weiterentwicklung des Kontext-Vektor-Ansatzes wird in *Kapitel 5.3* gesprochen.

Kapitel 6 stellt zunächst wichtige Konzepte und Frameworks vor, die bei der Implementierung der vorgestellten Ansätze verwendet wurden, bevor in *Kapitel 7* die Implementierung eines der vorgestellten Konzepte beschrieben wird. Neben der Effizienz stand bei der Umsetzung besonders die Einsetzbarkeit des Vefahrens in einer Produktionskette eines Workflow-Systems für historische Dokumente im Vordergrund.

In *Kapitel 8* werden die in dieser Arbeit vorgestellten Ansätze in unterschiedlichen Testszenarien evaluiert.

Das daran anschließende *Kapitel 9* beschreibt das Workflow-System zur Digitalisierung historischer Dokumente, welches in Kooperation mit dem Digitalisierungszentrum der Universität Würzburg entstanden ist und vergleicht es mit anderen, am Markt befindlichen Lösungen. Alleinstellungsmerkmal des Systems ist die Einbindung der zuvor vorgestellten und anderer Verfahren zur automatisierten Metadaten-Extraktion. Das Kapitel beschreibt, wie das System dadurch originalgetreuere Digitalisate produzieren kann, die Durchlaufzeit durch frühzeitige Erkennung von Produktionsfehlern minimiert, die Qualität der Reproduktionen maximiert und durch die explizite Metadaten-Verarbeitung komplexe Präsentationsformen erzeugen und vielfältige Nachnutzungsmöglichkeiten für Kooperationspartner bieten kann.

Kapitel 10 schließt mit einer Zusammenfassung der in dieser Arbeit erreichten Ziele und informiert über mögliche Erweiterungen, die in einzelnen Bereichen vorgenommen werden könnten.

2 Aufgabenstellung und Definitionen

Dieses Kapitel soll die Zielsetzung dieser Arbeit spezifizieren und stellt dazu die Definitionen häufig verwendeter Begriffe bereit.

2.1 Aufgabenstellung

Das vorherige Kapitel hat bereits angedeutet, dass Ansätze zur automatisierten *Named Entity Recognition* auf historischen Dokumenten mit den Besonderheiten der Domäne wie der hohen Schreibweisenvarianz zu kämpfen haben. Anstatt sich nun darauf zu beschränken, die Varianzen zu kompensieren, sollen in dieser Arbeit Ansätze untersucht werden, welche auch andere Informationsquellen in den Dokumenten zur Term-Klassifikation heranziehen.

Im Gegensatz zur oft nur konventionell vereinbarten Orthographie war in historischen Dokumenten der Satzbau und die Grammatik schon ähnlich restriktiv definiert wie in zeitgenössischen Dokumenten. Daraus folgt, dass die Wahrscheinlichkeit, mit der zwei Terme t_1, t_2 in direkter Abfolge in einem Dokument erscheinen, keinesfalls beliebig hoch ist, sondern eine spezifische (und damit messbare) Wahrscheinlichkeit hat.

Weiterhin bestehen Sätze hauptsächlich aus *Stopworten*. Stopworte sind hochfrequente Terme, die, abgesehen von ihrer linguistischen Funktion, nur wenig zur

Semantik des Satzes beitragen. Beispiele für Stopworte sind Präpositionen, Konjunktionen und Artikel. Ihr Anteil kann nach [Bird, Klein und Loper (2009)] in Abhängigkeit von der Domäne des untersuchten Dokuments bis zu 40 Prozent betragen. Aufgrund ihres häufigen Auftretens im Vergleich zu Nicht-Stopworten ist es jedoch sehr wahrscheinlich, dass sich ein Autor für eine bestimmte Schreibweise entschieden hat, wodurch die Stopworte eine viel höhere orthographische Konsistenz aufweisen als nur vereinzelt verwendete Begriffe.

In unserem Bestreben, die Semantik eines Werks automatisiert zu erfassen und zu digitalisieren, können Stopworte aufgrund ihrer hauptsächlich linguistischen Funktion natürlich offensichtlich vernachlässigt werden. Jedoch ist selbst in der komplementären Gruppe (die Nicht-Stopworte) nur eine Teilmenge der Terme von Interesse. Wenn man an Dokumente in Nachschlagewerken wie beispielsweise Wikipedia[1] denkt, sind dort üblicherweise nur bestimmte Terme (zumeist *Named Entities*) mit Verweisen auf weiterführende Dokumente versehen, beispielsweise Personennamen, Funktionen, Rollen oder Berufe, Orte und Zeitpunkte. Die hier vorgestellten Verfahren beschränken sich daher auf das automatisierte Erkennen von Named Entities, insbesondere Ortsangaben in historischen Dokumenten.

In dieser Arbeit werden Ansätze untersucht, welche sich die Information, die im Kontext eines Terms enthalten ist, zunutze machen können, um auf die Semantik des Terms schließen zu können. Das Problem, dass die Leistung von Klassifikatoren schon bei kleinen Änderungen der Domäne, auf der sie operieren, einbricht (siehe Definition *Named Entity Recognition* auf Seite 26), soll umgangen werden, da der Kontext der Terme, für die wir uns interessieren, häufig aus Stopworten mit hoher orthographischer Konsistenz besteht. Durch geschickte Auswertung der Charakteristiken der Stopworte im Kontext eines Terms und domänenspezifischer Heuristiken soll entschieden werden, ob er zu einer gesuchten Klasse gehört. Damit lässt sich auch auf syntaktisch schwierigen oder sogar beschädigten und insbesondere historischen Texten Named Entity Recognition betreiben.

[1] http://de.wikipedia.org

Die Nachnutzung der gewonnenen Metadaten ist dabei vielfältig: Die vorgestellten Ansätze sollen modular in ein Workflow-System für Digitalisierungsprozesse eingebunden werden, welches speziell für historische Dokumente wie Handschriften, Drucke und alte Landkarten entworfen wurde. In Abgrenzung zu derzeit üblichen Digitalisierungsverfahren ist die Erfassung und Auswertung von Metadaten expliziter Bestandteil des Digitalisierungsverfahrens. Die so gewonnenen strukturellen und semantischen Metadaten sollen verwendet werden, um den Anteil menschlicher Beteiligung bei der Metadaten-Erfassung in der Retro-Digitalisierung zu verringern. Aus den gewonnenen Metadaten können Heuristiken erzeugt werden, welche Produktionsfehler schon während der Digitalisierung aufzeigen können. Weiterhin ermöglichen die Metadaten die Verwendung von spezialisierten Präsentationswerkzeugen, welche dem Endnutzer ein mehr als bloße Bilddaten umfassendes und somit originalgetreueres Digitalisat präsentieren können. Das in dieser Arbeit vorgestellte Workflow-System soll die Vorteile einer Verarbeitungskette verdeutlichen, die Metadaten als integralen Bestandteil des Digitalisierungsprozesses behandelt.

2.2 Definitionen

Bevor weiter auf die grundlegenden Probleme eingegangen wird, die im Zusammenhang mit der Digitalisierung alter Dokumente auftreten können, sollen zunächst einige häufig verwendete Begriffe definiert werden.

Digitalisierung

Definition 1. Unter dem Begriff der Digitalisierung versteht man im Allgemeinen die Erstellung einer digitalen Repräsentation von Objekten, Bildern, Dokumenten oder akustischen Signalen. Bei diesem Vorgang werden kontinuierliche Größen in diskrete Werte überführt (so definiert beispielsweise in [*Duden* (2014)], weswegen die digitale Repräsentation immer nur eine Annäherung an das Original darstellt.

Für die Digitalisierung sprechen viele Vorteile: Die digitale Form eines Objekts kann einem breiten Publikum zugänglich gemacht werden, ohne dass das Original gefährdet wird (beispielsweise durch Abnutzung, Beschädigung oder Diebstahl). Digitale Objekte können maschinell analysiert, indexiert und verschlagwortet werden, um Nutzern präzise Antworten zu Suchanfragen zu liefern. Ein weiterer, wichtiger Aspekt der Digitalisierung ist sicherlich die Möglichkeit der (Langzeit-) Archivierung von Informationen. Bis dato gibt es keinen unendlich haltbaren Datenträger, weswegen digitale Informationen regelmäßig umkopiert oder redundant gespeichert werden müssen. Kopien einer digitalen Vorlage können ohne Qualitätsverlust erstellt werden, während ein analoges Medium während eines Kopiervorganges zumeist eine Qualitätsverminderung erfährt oder Gefahr läuft, beschädigt zu werden.

Retro-Digitalisierung

Definition 2. Die Retro-Digitalisierung stellt ein Spezialgebiet der Digitalisierung dar, welche sich auf die Erstellung von digitalen Repräsentationen historischer Dokumente, Handschriften und Landkarten spezialisiert. Bei dieser Art der Digitalisierung kommen Spezialscanner und -kameras zum Einsatz, die dem Umstand Rechnung tragen, dass die zu digitalisierenden Objekte oft hoch empfindlich und mit herkömmlichen bildgebenden Verfahren nicht zu erfassen sind. Um eine möglichst detailgetreue Annäherung an das Original zu erhalten, darf die Retro-Digitalisierung nicht mit der Erfassung der Bilddaten enden. Vielmehr müssen auch die logische und physikalische Struktur des Werkes (beispielsweise Unterteilung in Kapitel, Position von Inhaltsverzeichnissen) sowie die Semantik erfasst werden. Der Oberbegriff „Digitalisierung“ steht also für eine Erfassung möglichst vieler Aspekte eines Werkes und der Speicherung in geeigneten Archiv-Formaten.

Metadaten

Detaillierte Beschreibungen zu einer Ressource werden Metadaten genannt. In [Organization (2004)] wird der Begriff Metadaten wie folgt definiert:

> Metadaten sind strukturierte Informationen, die eine bestehende Informations-Quelle beschreiben, erklären, örtlich festlegen oder sie leichter verwendbar oder verwaltbar machen. Metadaten werden daher auch oft als Daten über Daten oder Informationen über Informationen bezeichnet.

[Organization (2004)] unterscheidet weiterhin zwischen drei Kategorien von Metadaten: *Beschreibende Metadaten* liefern Daten, die zur Identifikation und Beschreibung der Ressource verwendet werden können, wie beispielsweise der Autor und Titel, aber auch Inhaltsangaben. *Strukturelle Metadaten* geben Aufschluss über die Zusammensetzung eines Verbundobjekts aus mehreren Unterobjekten. Im Falle eines digitalisierten Buchs wäre hier die Information zu finden, welche Seiten zu Kapiteln zusammengebunden sind und aus welchen Kapiteln das Werk besteht. In der dritten Kategorie finden sich *administrative Metadaten*, in denen beispielsweise die Rechtesituation des Werkes erläutert oder Hinweise zur Archivierung gegeben werden können.

Die Retro-Digitalisierung, wie sie in der Digitalisierungsabteilung der Universitätsbibliothek der Universität Würzburg betrieben wird, macht es sich zum Ziel, neben der Erfassung des visuellen Aspekts eines Werks auch umfassende Metadaten bereitzustellen.

String

Diese Arbeit stellt Verfahren zur Klassifizierung von Strings vor.

Definition 3. Als String s werden endliche Zeichensequenzen über einem Alphabet Σ bezeichnet ($s \in \Sigma^*$).

Die Länge eines Strings s wird als die Länge der Zeichenfolge definiert und als $|s|$ notiert. Der *leere String* wird als ϵ bezeichnet und hat die Länge 0.

$s[i]$ bezeichnet das i-te Zeichen in der Zeichenfolge s mit $0 \leq i < |s|$.

Das Aneinanderfügen zweier Strings s und t mit $|s| = k$ und $|t| = l$ mit

$$st = s[0]s[1] \dots s[k-1]t[0]t[1] \dots t[l-1]$$

wird als *Konkatenation* definiert.

Satz

Im Laufe dieser Arbeit sollen Quelldokumente analysiert und einzelne Worte klassifiziert werden. Dazu wird das Quelldokument zunächst in Sätze zerlegt.

Definition 4. Ein Satz s ist eine endliche, nicht-leere Folge von Zeichen über dem Alphabet $\Sigma \backslash T$, T ist hierbei eine Menge von Satz-Trennzeichen.

Ein Quelldokument d lässt sich damit eindeutig in die Sätze s_i mit $1 \leq i \leq n$ zerlegen und es gilt: $d = t_0 s_1 t_1 \dots s_n t_n$ mit $t_0, t_n \in T^*$, $t_{1\dots n-1} \in T^+$ und $s_{1\dots n} \in (\Sigma \backslash T)^+$.

Wort

Nach der Zerlegung des Quelldokuments d in n Sätze s_i, $1 \leq i \leq n$ erfolgt die Zerlegung der einzelnen Sätze in Worte.

Definition 5. Analog zum Satz wird ein Wort als endliche, nicht-leere Folge von Zeichen über dem Alphabet $\Sigma \backslash U$ definiert, U sei eine Menge von Wort-Trennzeichen.

Ein Satz s lässt sich in die Worte w_j, $1 \leq j \leq k$ zerlegen und es gilt: $s = u_0 w_1 u_1 \dots w_k u_k$ mit $u_0, u_k \in U^*$, $u_{1\dots k-1} \in U^+$ und $w_{1\dots k} \in (\Sigma \backslash U)^+$.

n-Gramm

Definition 6. [Dunning (1994), Cavnar und Trenkle (1994)] definieren ein n-Gramm (auch q-Gramm) als einen String der Länge n (q).

Ein String s mit $|s| \geq n$ kann in $(|s| - n + 1)$ n-Gramme zerlegt werden, indem man ein Zeichen-Fenster der Breite n über s legt und es jeweils um ein Zeichen weiter verschiebt, um das nächste n-Gramm zu erzeugen.

Term

Definition 7. Je nach Granularität, mit der die Sätze zerlegt werden, bezeichnet der Begriff Term entweder ein Wort oder ein n-Gramm. Nachfolgend steht der Begriff Term somit für die kleinsten semantischen Einheiten, deren Eigenschaften in den vorgestellten Verfahren zur Klassifizierung herangezogen werden sollen.

Kontext

Im Verlauf dieser Arbeit wird häufig der Begriff Kontext verwendet. In den Sprachwissenschaften wird der (verbale) Kontext als der umgebende Text oder das umgebende Gespräch eines Ausdrucks definiert. Analog dazu sei in dieser Arbeit definiert:

Definition 8. Sei w ein Term innerhalb eines Satzes s über dem Alphabet $(\Sigma \backslash T)$, wobei T eine Menge von Satz-Trennzeichen ist. Dann enthält s den Kontext zu w und ist eine endliche, nicht-leere Folge von Zeichen:

$$s = r_l c_l w c_r r_r$$

mit $r_l, r_r \in (\Sigma \backslash T)^*$ und

$$context(w) = c_l c_r$$

mit $|context(w)| > 0$

Somit ist der Kontext eines Terms w der ihn umgebende textuelle Abschnitt, wobei die Granularität frei bestimmt werden kann. Häufig umfasst er jedoch beispielsweise den Satz, in dem der Term sich befindet, oder besteht aus einem festen Fenster von Termen oder Zeichen um w herum.

Named Entity Recognition

Unter dem Begriff *Named Entity Recognition* (NER) versteht man den Teilbereich des Information Retrievals, der sich damit befasst, Elemente in einem Text in vordefinierte Kategorien zu klassifizieren, beispielsweise Ortsangaben (Toponyme), Personennamen, Zeitangaben oder Rollen (wie beispielsweise Berufe).
Die NER ist ein wichtiger Vorverarbeitungsschritt, um Einheiten in Texten zu identifizieren, die für dessen Semantik von hoher Bedeutung sind. Named Entities sind häufig Begriffe, nach denen ein Nutzer in Suchmaschinen suchen würde, oder die sich für Querverweise in weiterführende Quellen eignen. Grundlegend wird zwischen Ansätzen unterschieden, die die Klassifikation aus statistischen Modelle ableiten und solchen, die eine Grammatik der zugrundeliegenden Sprache verwenden. Die Genauigkeit von NER-Methoden erreicht für beispielsweise die englische Sprache ein quasi-menschliches Niveau (siehe [Marsh und Perzanowski (1998)]), allerdings sind dazu im Vorfeld aufwändige Trainingsphasen und eine intensive Beteiligung menschlicher Experten notwendig, um die verwendeten Modelle auf die Domäne zu spezialisieren.
Nach [Poibeau und Kosseim (2001)] sind aber selbst aktuelle Ansätze sehr anfällig gegenüber kleinsten Veränderungen in der Domäne, sodass selbst bei gleichbleibender Sprache eine Veränderung der Spezifität eines Texts (wissenschaftliche Abhandlung im Vergleich zu prosaischer Erzählung) oder die Verwendung eines Dialekts ausreicht, die Leistung eines Klassifikators einbrechen zu lassen.

Markov Eigenschaft

Definition 9. Nach [Durrett (2010)] besitzt ein stochastischer Prozess die *Markov Eigenschaft* genau dann, wenn die bedingte Wahrscheinlichkeitsverteilung zukünftiger Zustände des Prozesses unabhängig von der Sequenz vergangener Zustände ist und nur vom aktuellen Zustand bestimmt wird. Ein Prozess, der diese Eigenschaft besitzt, wird Markov-Prozess oder Markov-Kette genannt.
Für eine endliche Zustandsmenge $S = s_1, \ldots, s_m$ und der diskreten Zeit $t = 0, 1, 2, \ldots$ ist ein stochastischer Prozess $X = (X_t)_{t=0,1,\ldots}$ mit Werten in S eine Markov-Kette mit der Eigenschaft

$$\begin{aligned} &P(X_{t+1} = s_{j_{t+1}} | X_t = s_{j_t}, X_{t-1} = s_{j_{t-1}}, \ldots, X_0 = s_{j_0}) \\ &= P(X_{t+1} = s_{j_{t+1}} | X_t = s_{j_t}) \end{aligned}$$

Metrik

Definition 10. Eine *Metrik* oder *Distanzfunktion* beschreibt nach [Arkhangelskii und Fedorchuk (1990)] den Abstand zwischen zwei Elementen einer Menge. Eine Menge mit einer Metrik wird als *metrischer Raum* bezeichnet.

Eine Metrik auf einer Menge S ist demnach eine Funktion

$$dist : S \times S \to \mathbb{R}$$

Für alle $s, t, u \in S$ müssen folgende Bedingungen gelten:

1. $dist(s, t) \geq 0$ (Nicht-Negativität)
2. $dist(s, t) = 0 \Leftrightarrow s = t$ (Gleichheit von Ununterscheidbaren)
3. $dist(s, t) = dist(t, s)$ (Symmetrie)
4. $dist(s, u) \leq dist(s, t) + dist(t, u)$ (Dreiecksungleichung)

3 Verwandte Arbeiten

In diesem Kapitel soll eine Übersicht über Verfahren der Named Entity Recognition (NER) (wie definiert in Abschnitt 2.2) gegeben werden, die sich die Information im Kontext eines Terms für eine Klassifikation zu Nutze machen.

3.1 Literatur

Zahlreiche Arbeiten haben sich mit der Information beschäftigt, die im Kontext eines Terms zu finden ist: Nach [Miller und Charles (1991)] korreliert die Austauschbarkeit zweier Terme in einem gegebenen Kontext mit ihrer semantischen Ähnlichkeit. Das bedeutet, je einfacher zwei Terme in den jeweiligen Kontexten austauschbar sind, in denen sie auftauchen, desto höher ist die Wahrscheinlichkeit, dass sie eine ähnliche Bedeutung haben. Eine statistische Analyse der Kontexte zweier Terme kann also Ausschluss über den Grad ihrer Ähnlichkeit liefern.
Viele Verfahren machen sich daher auch die Information, die im Kontext eines Terms steckt, zu Nutze: [Gauch, Wang und Rachakonda (1999)] schlagen beispielsweise vor, basierend auf der Analyse von Term-Nachbarschaften eine Suchanfrage um zusätzliche Terme zu erweitern. [Billhardt u. a. (2002)] analysieren Term-Nachbarschaften, um auf Beziehungen und Abhängigkeiten zwischen Termen schließen zu können. [Schütze (1992)] hingegen analysiert die Komposition des Kontexts eines Terms, um Kontext-Vektoren zu erzeugen, mit deren Hilfe dann mehrdeutige Terme voneinander abgegrenzt werden können. Da dieser Ansatz allerdings lediglich auf die Zusammensetzung des Kontexts eingeht, werden statistisch verwertbare Effekte durch Satzbau und Grammatik der zugrunde liegenden Sprache vernachlässigt. Daher soll der Ansatz im weiteren Verlauf der Arbeit aufgegriffen und

zu einem Klassifikationsverfahren verallgemeinert werden, welches umfassendere kontextuelle Eigenschaften auswertet. Dazu wird der in [Schöneberg (2010)] und [Schöneberg und Müller (2012)] beschriebene Ansatz zur Erzeugung eines hochdimensionalen Kontext-Vektors verfolgt. Dabei kommt das Vector Space Model zum Einsatz, wie es in [Manning, Raghavan und Schütze (2008)] und [Baeza-Yates und Ribeiro-Neto (1999)] vorgestellt wird.

Viele Verfahren im Bereich der NER basieren auf Conditional Random Fields (beschrieben in Definition 12). Auch diese machen sich die Information im Kontext des zu klassifizierenden Terms zu Nutze, beispielsweise in den in [Kluegl u. a. (2012)] und [Kluegl u. a. (2013)] beschriebenen Verfahren, welche die strukturellen Merkmale des Term-Kontexts in dessen Klassifikation einfließen lassen. In [Müller (2012)] wurde ein auf CRFs basierendes Verfahren evaluiert, welche neben heuristischen auch kontextuelle Informationen zur Klassifikation von Toponymen in historischen Texten verwendet hat.

3.1.1 Historische Texte

Historische Texte stellen eine besondere Herausforderung für automatisierte Ansätze dar. In [Piotrowski (2012)] wird ein allgemeiner Überblick über computergestützte Verfahren aus den Bereichen NER und NLP gegeben. Als besonders schwerwiegender Einfluss wird die hohe Schreibweisenvarianz der Quelldokumente genannt. Einerseits wird diese durch die nur per Konvention vereinbarte Rechtschreibung verursacht (wie in Abschnitt 1.1 bereits beschrieben), andererseits sind auch die OCR-Verfahren, mit denen die Texte in eine digitale Form überführt werden, eine nachgelagerte Fehlerquelle. [Rodriquez u. a. (2012)] stellt gängige NER-Verfahren für aus OCR gewonnene Texte vor und betont die stark domänenabhängige Leistung der Klassifikationsläufe.

[Hauser u. a. (2007)] unterscheiden zur Überwindung der syntaktischen Schwierigkeiten zwischen drei verschiedenen Ansätzen: Einerseits werden Verfahren ge-

nannt, die auf *domänenspezifischen Wörterbüchern* basieren. Diese sind aber nur mit hohem zeitlichen Aufwand zu erstellen. Weiterhin werden regelbasierte Ansätze genannt, die zu einem neusprachlichen Wort mögliche historische Schreibweisen generieren, oder eine historische Wortform auf eine moderne abzubilden versuchen. Zuletzt werden Ansätze angesprochen, die zwei Terme anhand ihrer Ähnlichkeit oder Distanz zuordnen.

Auch in der Domäne der historischen Dokumente werden Verfahren angewendet, welche die Information, die im Kontext des zu klassifizierenden Terms steckt, während des Klassfikationsvorgangs hinzuziehen. [Bollmann, Petran und Dipper (2011)] stellen beispielsweise ein regelbasiertes Verfahren vor, bei dem Informationen aus dem Kontext eines Terms verwendet werden, um Bezüge zwischen verwandten Wortformen herstellen zu können. [Smith und Crane (2001)] ziehen die kontextuellen Informationen eines Terms heran, um zwischen Orts- und Personennamen zu unterscheiden. Sie beschränken sich dazu aber auf eine zuvor definierte Domäne und verwenden externes, domänenspezifisches Wissen (beispielsweise Datenbanken). Weiterhin verweisen sie in ihrer Arbeit auf die Wichtigkeit bestimmter Named Entities wie Personen, Rollen, Datums- und Ortsangaben.

In historischen Texten ist insbesondere die Erkennung von verschachtelten Entities ein schwieriges Problem: Bei verschachtelten Entities handelt es sich um Komposita, deren Bestandteile eine andere Klassifikation besitzen als der zusammengesetzte Ausdruck.

Herzog Maximilian von Bayern

Abbildung 3.1: Verschachtelte Entities.

Nur anhand des Kontexts kann entschieden werden, wie ein Kompositum zu klassifizieren ist. Abbildung 3.1 zeigt beispielsweise die Phrase *„Herzog Maximilian von Bayern“*, die als *Rolle* zu klassifizieren wäre, allerdings auch die Named Entities *„Maximilian von Bayern“* (*Name*) und *„Bayern“* (*Toponym*) enthält. [Byrne (2007)] demonstrieren ein Verfahren, um verschachtelte Entities in historischen

Texten zu erkennen. Dazu ist jedoch eine komplexe Verkettung von Vorverarbeitungsschritten notwendig. Die korrekte Klassifikation von verschachtelten Entities stellt eine weitere wichtige Verwertungsmöglichkeit kontextueller Information dar.

Nationale Einrichtungen wie die *Deutsche Nationalbibliothek*[1] unterstreichen die Wichtigkeit dieser Ansätze, da in der *Gemeinsamen Normdatei (GND)* (siehe [Nationalbibliothek (2014)], [*GeoNames* (2014)]) explizit auch historische Schreibweisenvarianten zu neusprachlichen Einträgen für beispielsweise Ortsnamen erfasst werden sollen.

Dieses Kapitel hat die Wichtigkeit der im Kontext eines Terms enthaltenen Information illustriert und einen Überblick über den aktuellen Stand der Forschung gegeben. Nachfolgend sollen nun Verfahren demonstriert werden, die den angesprochenen Schwierigkeiten, die die Domäne der historischen Texte mit sich bringt, begegnen, indem zur Klassifikation von Termen die Information analysiert wird, die im (orthographisch konsistenteren) Kontext eines Terms enthalten ist.

[1] www.dnb.de

4 Grundlagen kontextueller Analysen

Bevor in den nachfolgenden Sektionen eine Übersicht über gängige Verfahren, die sich die Information aus dem Kontext zu Nutze machen, gegeben werden kann, werden zunächst einige bekannte Verfahren aus der Named Entity Recognition vorgestellt.

4.1 Hidden Markov Model

Viele Verfahren, die erfolgreich in der Spracherkennung, Computerlinguistik oder Bioinformatik (beispielsweise zur Vorhersage von Proteinfaltungen) eingesetzt werden, basieren auf *Hidden Markov Models* (HMMs).

Definition 11. In [Rabiner (1989)] wird ein Hidden Markov Model als ein stochastisches Modell definiert, welches sich durch zwei Zufallsprozesse beschreiben lässt. Der erste Zufallsprozess entspricht dabei einer Markov-Kette, deren Zustände von außen jedoch nicht sichtbar (daher *hidden*) sind. Zusätzlich werden von einem zweiten Zufallsprozess beobachtbare *Emissionen* generiert, die von dem Zustand des ersten Prozesses abhängen.

Das Modell formuliert zwei Annahmen:

- Die *Markov-Annahme* besagt, dass ein Übergang in den Nachfolgezustand im ersten Prozess immer nur von dessen aktuellem Zustand abhängt und nicht von den vorhergehenden (siehe Definition 9).

- Weiterhin besagt die *Unabhängigkeits-Annahme*, dass eine Emission des zweiten Prozesses zu jedem Zeitpunkt lediglich vom Zustand des ersten Prozesses abhängt und unabhängig von vorherigen Emissionen und Zuständen ist.

Da die Zustände des ersten Prozesses allerdings versteckt sind, können dessen Zustandsübergänge nur aus den Emissionen des zweiten Zufallsprozesses abgeleitet werden, welche in Abhängigkeit vom Zustand des ersten Zufallsprozess mit einer gewissen Wahrscheinlichkeit erfolgen. Abbildung 4.1 zeigt schematisch den Aufbau eines HMMs sowie eine beispielhafte Emissions-Sequenz. Eine typische Aufgabe im Zusammenhang mit HMMs ist es, aus einer Sequenz von beobachteten Emissionen zu einer Vorhersage über die wahrscheinlichste Sequenz von versteckten Zuständen des ersten Prozesses zu kommen. Die Zustände entsprechen dabei Klassifikationen (beispielsweise „ist Ort“ im Gegensatz zu „ist kein Ort“). Dieses Problem lässt sich mit Hilfe des *Viterbi-Algorithmus* [Forney (1973)] lösen.
Für unsere Zielsetzung, Terme anhand ihres Kontexts zu klassifizieren, ist es erforderlich, die Übergangswahrscheinlichkeiten des ersten Prozesses in einem HMM derart zu modifizieren, dass es eine gegebene Emissions-Sequenz am wahrscheinlichsten generiert. Mit dem *Baum-Welch-Algorithmus* (beschrieben beispielsweise in Manning und Schütze (1999)) und einem Traningscorpus kann dieses geleistet werden. Nachdem das HMM in dieser Art ausreichend anhand von Beispielen trainert wurde, kann das Modell dazu verwendet werden, auch unbekannte Sequenzen zu klassifizieren.

In der vorliegenden Arbeit sollen die Ansätze, die auf HMMs basieren, jedoch nicht weiter verfolgt werden. Stattdessen wird nachfolgend ein verwandtes Modell vorgestellt, das Conditional Random Field, da es einige Eigenschaften besitzt, die es für den vorliegenden Anwendungsfall attraktiver machen.

4.2 Conditional Random Field

Ähnlich wie das HMM ist auch das *Conditional Random Field* (CRF) ein Wahrscheinlichkeitsmodell, welches erfolgreich in Problemstellungen der Named Entity

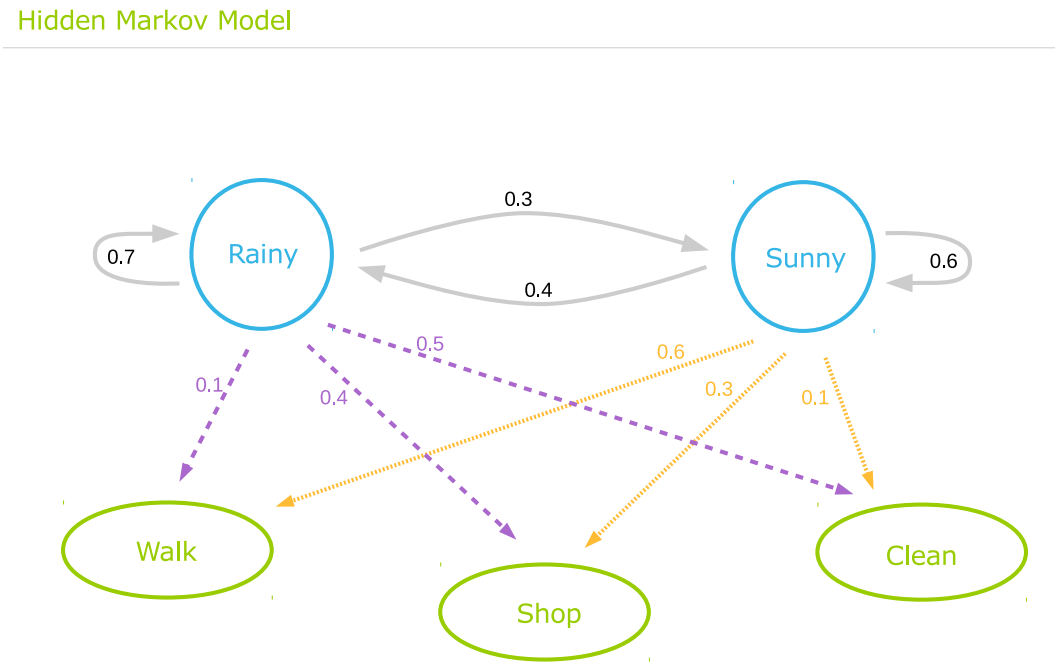


Abbildung 4.1: Allgemeine Funktionsweise eines Hidden Markov Models.

Recognition, Natural Language Processing, Bioinformatik und Computer Vision eingesetzt wird. Abbildung 4.2 zeigt den schematischen Aufbau eines CRFs.

Definition 12. Ein CRF ist ein ungerichtetes, graphisches Modell. [Lafferty, McCallum und Pereira (2001)] definieren ein CRF als einen Graphen $G = (V, E)$ über V, einer Menge von Knoten und E, einer Menge von Kanten. Sei X eine Menge von Beobachtungen. Weiterhin sei $Y = (Y_v)_{v \in V}$ eine Menge von Zustandsvariablen, so dass jede Zustandsvariable einem Knoten aus V entspricht. (X, Y) sei dann ein CRF, wenn die Zufallsvariablen Y_v die Markov-Eigenschaft (nach Definition 9) besitzen:

$$p(Y_v|X, Y_w, w \neq v) = p(Y_v|X, Y_w, w \sim v)$$

wobei $w \sim v$ bedeutet, dass w und v im Graphen benachbart sind. CRFs modellieren die bedingte Wahrscheinlichkeit von $p(Y|X)$.

Anders als ein HMM modelliert ein CRF Zustandsübergänge durch eine unbeschränkte Anzahl an Feature-Funktionen, die in Abhängigkeit von der Beobachtungssequenz von Zustand zu Zustand variieren können. Die Feature-Funktionen können dabei mit unterschiedlicher Granularität auf Eigenschaften der Beobachtungssequenz eingehen und auf vergangene und zukünftige Teile oder die komplette Beobachtungssequenz zugreifen. In einem HMM ist im Gegensatz dazu der Zustandsübergang durch konstante Wahrscheinlichkeiten in Abhängigkeit von der aktuellen Beobachtung definiert.
CRFs erlauben durch die Verwendung der Feature-Funktionen eine komplexe Analyse der Beobachtungssequenz. Die Zustandsvariablen aus Y können als „Label“ interpretiert werden, sodass eine Beobachtungssequenz X durch die entsprechende Abfolge von Zuständen Y_i klassifiziert werden kann. Wie zuvor auch schon das HMM muss ein CRF allerdings zunächst trainiert werden, bevor es zur Klassifikation unbekannter Sequenzen verwendet werden kann: Dazu werden Beobachtungssequenzen benötigt, für die die korrekte Abfolge von „Labels“ bekannt ist. Anhand dieser Beispiele werden dann Algorithmen wie der Gradientenabstieg (beschrieben beispielsweise in [Snyman (2005)]) oder Quasi-Newton-Verfahren wie Limited-memory BFGS [Andrew und Gao (2007)] verwendet, um die Fehlerrate bei der Label-Vorhersage des zu trainierenden CRFs zu reduzieren. Ein mit einer ausreichenden Menge von Beispielen trainiertes CRF kann dann zur Klassifikation unbekannter Sequenzen verwendet werden.

4.2.1 Feature Funktionen

Der Zustand Y_i, in dem sich ein CRF zu einem gegebenen Zeitpunkt i befindet, ist abhängig von einer festen Anzahl von Feature-Funktionen der Form $f(i, Y_{i-1}, Y_i, X)$. Das Modell weist allen Feature-Funktionen ein Gewicht zu und kombiniert sie, um die Wahrscheinlichkeit eines Wertes für Y_i zu berechnen. Für die Klassifikation der Beobachtung sind Feature-Funktionen damit von zentraler Bedeutung. Wie eingangs erwähnt können die Feature-Funktionen jederzeit auf die gesamte Beobachtungssequenz zugreifen und mit unterschiedlicher Granularität auf spezifische Eigenschaften eingehen. Diese Voraussetzungen machen das CRF zu einem viel-

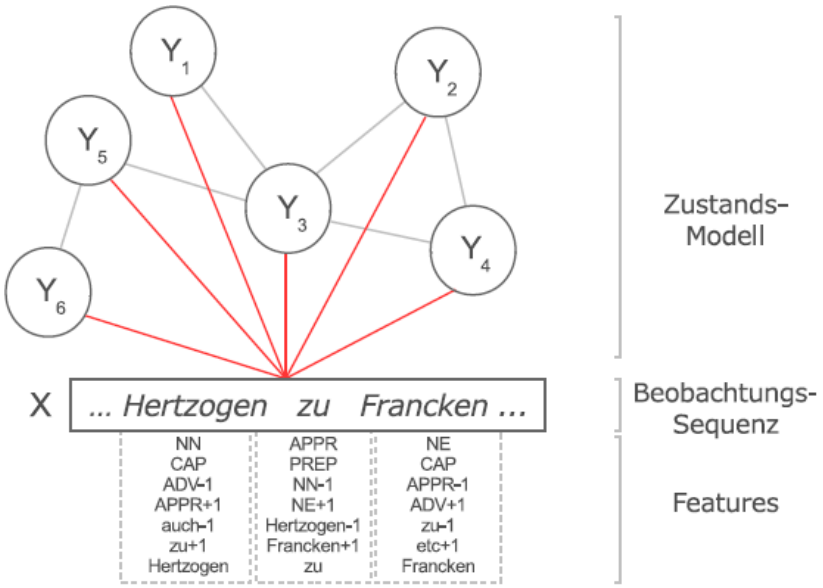


Abbildung 4.2: Conditional Random Field und Auszug aus Beobachtungssequenz X.

versprechenden Kandidaten für einen Einsatz als Klassifikator, der Informationen des Kontexts berücksichtigen kann. Im weiteren Verlauf dieser Arbeit sollen sowohl statistische als auch heuristische Feature-Funktionen vorgestellt werden und die Eignung von auf CRFs basierenden Ansätzen zur Klassifikation von Termen in historischen Texten durch Kontext-Evaluation untersucht werden.

4.3 Vektorraum-Retrieval

Das Vektorraum-Retrieval ist ein allgemeines Verfahren (ursprünglich in [Baeza-Yates und Ribeiro-Neto (1999)] beschrieben), in dem Informationen Punkte in einem hochdimensionalen, metrischen Vektorraum darstellen. Zwei Informations-Punkte sind dann mit Hilfe einer Abstandsfunktion miteinander vergleichbar (siehe Abschnitt 5.2.5 - *Distanzmetriken*); komplexere Eigenschaften können als Vektoren in diesem Informationsraum ausgedrückt werden.
Etabliert hat sich das Vektorraum-Retrieval im Bereich von Suchmaschinen-Anfragen, so zum Beispiel bei der Identifikation von relevanten Dokumenten für eine

gegebene Suchanfrage oder bei der anschließenden Sortierung der Trefferlisten. Dabei werden Dokumente als Vektoren über einem Suchraum repräsentiert, dessen Dimensionen je nach Anwendungsfall beispielsweise durch die Menge aller möglichen Schlagworte für Dokumente, Terme oder Phrasen definiert sind.
Die Komponenten eines Dokumenten-Vektors drücken dann aus, wie relevant das Dokument für den jeweiligen Term, das jeweilige Schlagwort oder die jeweilige Phrase ist. Die Ähnlichkeit oder Relevanz eines Dokuments im Bezug auf die Suchanfrage wird anhand einer Distanzmetrik ermittelt. [Manning, Raghavan und Schütze (2008)] beschreiben beispielsweise eine bekannte Metrik für Vektorraum-Vergleiche, das Cosinus-Maß, welches den Winkel zwischen Anfrage- und Dokument-Vektor misst; parallele Vektoren (oder solche, die nur in einem geringen Winkel davon abweichen) werden als ähnlich oder wichtig angesehen. In Abschnitt 5.2.5 wird dieses Verfahren genauer beschrieben.
Da sich ganz allgemein beliebige Eigenschaften als Vektoren über dem Informationsraum ausdrücken lassen, soll in den nachfolgenden Kapiteln untersucht werden, ob ein Ansatz, der die im Kontext eines Terms enthaltenen Informationen extrahiert, diese als Vektoren über dem Informationsraum speichert und über Distanzmetriken eine Klassifikation unbekannter Terme versucht, sich als für unsere Problemstellung geeignet herausstellt.

Nach diesem Überblick über prominente Verfahren der NER werden im folgenden Kapitel drei unterschiedliche Ansätze zur Klassifizierung von Termen in historischen Texten vorgestellt.

5 Semantische Analyse durch Kontext-Auswertung

In den nachfolgenden Abschnitten sollen drei Ansätze vorgestellt werden, die das Ziel haben, Metadaten wie Ortsnamen aus historischen Texten zu extrahieren. Insbesondere soll dabei auf die Information zurückgegriffen werden, welche sich im Kontext des zu klassifizierenden Terms befindet.

5.1 Toponymbestimmung in historischen Dokumenten mit CRFs

Der erste Ansatz, der vorgestellt werden soll, beruht auf Conditional Random Fields (CRFs), wie sie in Definition 12 beschrieben wurden. Ziel des Verfahrens ist die Identifikation von *Toponymen* (Eigennamen topographischer Gegenstände wie zum Beispiel Länder, Städte, Gemarkungen) unter Verwendung der Informationen, die im Kontext des zu klassifizierenden Terms enthalten sind. Die gefundenen Toponyme sollen nach erfolgreicher Beendigung des Verfahrens in ein Format gebracht werden, welches die Weiterverarbeitung in nachgeschalteten Verarbeitungsschritten ermöglicht. Zur genauen Aufgabenstellung und Zielsetzung finden sich detailliertere Beschreibungen in [Müller (2012)].
Dieses Verfahren stellt einen hybriden Ansatz zwischen stochastischem Modell und einem regelbasierten Verfahren dar.

5.1.1 Konzept

Das Verfahren zur Toponymbestimmung in historischen Texten unter Verwendung von CRFs ist in Abbildung 5.1 dargestellt: Als Eingabe wird zunächst der zu untersuchende Text erwartet. Dieser muss in einem Vorverarbeitungsschritt mit *Features* annotiert werden.

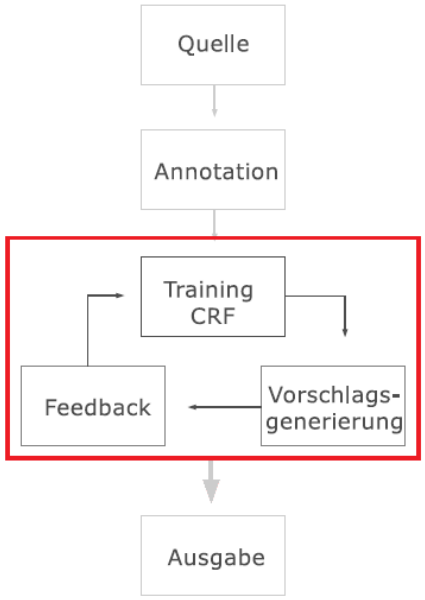


Abbildung 5.1: Konzept der Toponymbestimmung mit CRFs.

Dabei werden die in Abschnitt 4.2.1 angesprochenen Feature-Funktionen verwendet. Unter Anderem wurden folgende Feature-Funktionen bei diesem Ansatz verwendet:

- **POS-Feature** Diese Feature-Funktion versieht jeden Term der Beobachtungssequenz mit seiner Part-of-Speech (POS) Funktion. Dabei kam der TreeTagger [*TreeTagger* (2014)] zum Einsatz.
- **Kontext-Feature** Dieses Feature versieht jeden Term mit Co-Occurrence Informationen, indem benachbarte Terme erfasst werden.

- **Suffix-Feature** Diese Heuristik untersucht Terme auf typische Gemarkungs-Suffixe wie „-burg“, „-gau“, „-wald“ oder „-platz“.
- **Großbuchstaben-Feature** Diese Heuristik wird verwendet, da Ortsbezeichner häufig mit Großbuchstaben beginnen.
- **Toponym-Feature** Diese Feature-Funktion zeichnet typische Toponym-Indikatoren aus, beispielsweise „Kloster“, „Balley“, „Stadt“, „Amt“.

Abbildung 4.2 stellt die mit Features annotierte Beobachtungssequenz schematisch dar. Die Beobachtungssequenz wird beim anschließenden Training des CRFs ausgewertet und die Übergangswahrscheinlichkeiten zwischen den Zuständen angepasst. Im vorliegenden Verfahren wurde dabei das in Java geschriebene Tool *Mallet*[1] verwendet. In der Trainingsphase werden mit Hilfe eines Gradientenabstiegsverfahrens die Parameter des Modells optimiert [*Mallet Optimization* (2014)]. Mit dem trainierten CRF können nun auch unbekannte Beobachtungssequenzen klassifiziert werden. Anwenderfeedback zur Klassifikation kann genutzt werden, um das Modell in weiteren Trainingsiterationen zu verbessern.

5.1.2 Evaluation

In [Müller (2012)] wird die Leistungsfähigkeit dieses Ansatzes detailliert bewertet: Es stellt sich heraus, dass der beschriebene Ansatz geeignet ist, um in historischen Texten Toponyme zu identifizieren und die gefundenen Metadaten für eine Nachnutzung aufzubereiten.

Ziel dieser Arbeit ist jedoch nicht nur die Evaluation von Methoden zur Metadaten-Extraktion in historischen Texten, sondern auch die Verwendbarkeit der Ansätze als automatisierte Module in einem Workflow-System zu untersuchen. Abbildung 5.2 zeigt allerdings deutlich, dass mehrere Iterationen des Zyklus aus Nutzer-Feedback und erneutem Training des zugrundeliegenden Modells notwendig sind, um die Klassifikationsqualität zu steigern. Weiterhin stellen viele der zum Einsatz

[1] http://mallet.cs.umass.edu/

kommenden Feature-Funktionen spezifisches Wissen für eine Domäne dar, die bei einem Wechsel der Domäne vermutlich ungeeignet wären.

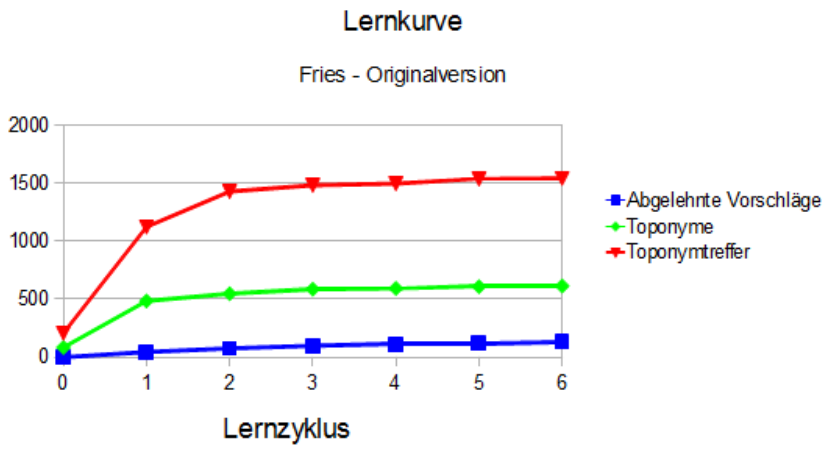


Abbildung 5.2: Verbesserungen der Ergebnisse über mehrere Trainingsiterationen. Quelle: [Müller (2012)]

Eine Verwendung dieses Ansatzes im Rahmen eines Workflow-Systems zur automatisierten Metadaten-Extraktion scheidet damit zunächst aus. Nachfolgend sollen Ansätze vorgestellt werden, welche ohne domänenspezifisches Wissen und Nutzerfeedback eine Extraktion von Metadaten vornehmen können.

5.2 Kontext-Vektor Retrieval in historischen Dokumenten

Analog zum prominenten Beispiel aus [Baeza-Yates und Ribeiro-Neto (1999)] soll beim Kontext-Vektor Retrieval versucht werden, die kontextuellen Eigenschaften von Termen als Vektoren in einem Vektorraum darzustellen. Über geeignete Abstandsfunktionen soll dann ermittelt werden, welche Terme sich kontextuell ähneln und gleich klassifiziert werden sollten.

Data : Kontext-Vektor Datenbank DB_v, Liste von Referenz-Termen r,
Ähnlichkeits-Schwellwert d
Result : Liste von Kontext-Vektoren v_r, die r ähneln

begin
  $v_r \longleftarrow List()$;
  /* Referenz-Vektor für r ermitteln */
  $\vec{c_r} \longleftarrow getContextVector(r)$;
  /* für jeden Kontext-Vektor in DB_v */
  forall the $\vec{v}$ *in* DB_v **do**
    /* Ähnlichkeit ermitteln */
    $sim \longleftarrow similarity(\vec{v}, \vec{c_r})$;
    /* Falls ähnlich genug ... */
    if $sim \geq d$ **then**
      /* ...aktuellen Vektor zu Ergebnis-Liste hinzufügen */
      $v_r \longleftarrow add(v_r, \vec{v})$
    end
  end
  return v_r;
end

Algorithmus 1 : Kontext-Vektor Retrieval

5.2.1 Konzept

Algorithmus 1 zeigt zunächst das allgemeine Konzept hinter der Klassifikation durch Kontext-Vektoren: Das Ziel ist die Identifikation aller Terme, die zu einer gesuchten Klasse gehören. Als Eingabe wird dazu eine Kontext-Vektor Datenbank DB_v vorausgesetzt, die eine Vektor-Repräsentation der kontextuellen Eigenschaften jedes Terms eines Text-Corpus enthält, eine Liste von Referenz-Termen r, die repräsentative Instanzen der gesuchten Klasse enthält und anhand derer die Klassifikation vorgenommen werden soll, sowie ein Schwellwert d, der angibt, ab welcher

Ähnlichkeit zwei Kontext-Vektoren $\vec{a}, \vec{b}$ als ähnlich genug angesehen werden sollen. Der Algorithmus liefert bei erfolgreicher Beendigung eine Liste, die diejenigen Kontext-Vektoren enthält, die die geforderte Ähnlichkeit zu r besitzen.

Um den Algorithmus umsetzen zu können, müssen jedoch noch einige seiner Bausteine definiert werden:

1. Das Ziel des Algorithmus ist es, Terme anhand ihrer kontextuellen Eigenschaften zu klassifizieren. Abschnitt 5.2.2 legt zunächst dar, was unter dem *Kontext* eines Terms verstanden wird und wie er zur weiteren Analyse extrahiert werden kann.

2. Nachdem festgelegt wurde, was der Kontext eines Terms ist, gilt es dessen Eigenschaften zu analysieren. Die Eigenschaften, die in Abhängigkeit vom untersuchten Text-Corpus, seiner Domäne und dem Klassifikations-Ziel variieren können, werden in Abschnitt 5.2.3 vorgestellt.

3. Bevor die kontextuellen Eigenschaften eines Terms als Vektor dargestellt werden können, muss zunächst festgelegt werden, wie der Vektorraum aufgebaut ist. In Abschnitt 5.2.4 wird die Struktur des Vektorraumes erläutert.

4. Nachdem der Vektorraum definiert wurde, können kontextuelle Eigenschaften als Vektoren über diesem Raum dargestellt werden. Der Algorithmus zur Erzeugung eines Kontext-Vektors für einen gegebenen Term und dessen Struktur wird in Abschnitt 5.2.4 erläutert.

5. Der letzte Baustein, der für die Klassifikation von Termen anhand ihrer Kontext-Vektoren notwendig ist, ist das Ähnlichkeitsmaß, anhand dessen beurteilt werden soll, ob zwei Terme die gleiche Klassifikation erhalten sollen. Abschnitt 5.2.5 stellt das verwendete Verfahren sowie Vorzüge und Nachteile für den vorliegenden Anwendungsfall dar.

5.2.2 Extraktion des Kontexts

Analog zur Definition 8 (siehe Seite 25) wird der Kontext eines Terms als der ihn umgebende textuelle Abschnitt aufgefasst.

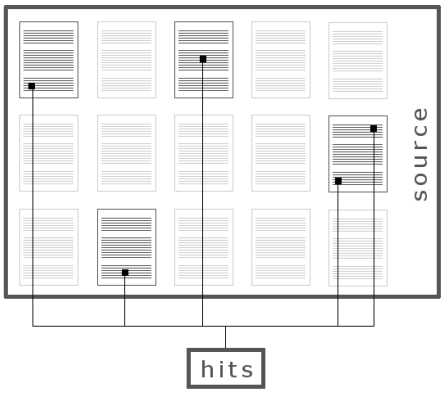


Abbildung 5.3: Lokalisierung aller Vorkommen eines Wortes im Text-Corpus.

Die Schnittstelle *Contexter* definiert die Methode `getContext()`, die zu einem gegebenen Term diejenigen Worte (nach Definition 5 auf Seite 24) liefert, die seinen Kontext ausmachen. Kapitel 7 stellt einige mögliche Implementierungen dieser Schnittstelle vor. Es obliegt dem Anwender, eine für den untersuchten Text-Corpus, dessen Domäne und für das Klassifikationsziel angepasste Implementierung zu wählen. Abbildung 5.4 stellt die Arbeitsweise der Contexter-Schnittstelle schematisch dar.

5.2.3 Feature-Module

Die zentrale Aufgabe im Klassifikationsprozess ist die Analyse der kontextuellen Eigenschaften eines Terms. Diese Analyse wird von den *Feature-Modulen* geleistet. Die Feature-Module erfüllen die zu den Feature-Funktionen aus dem am Anfang des Kapitels beschriebenen Ansatz analoge Funktion: Ein Feature-Modul nimmt

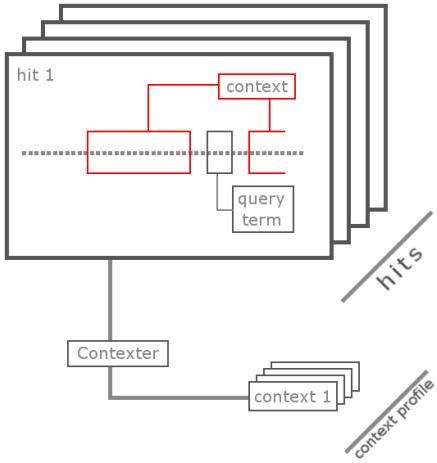


Abbildung 5.4: Arbeitsweise der Contexter-Schnittstelle.

den Kontext eines Terms entgegen, der von dem im vorherigen Abschnitt definierten Contexter extrahiert wird, und liefert eine Menge von *Features* zurück, die die über den Kontext gesammelten Informationen codieren. Abbildung 5.5 illustriert das Prinzip. Je nach verwendetem Text-Corpus und dem Klassifikationsziel müssen die Feature-Module dabei auf unterschiedliche Aspekte des Kontexts eingehen.

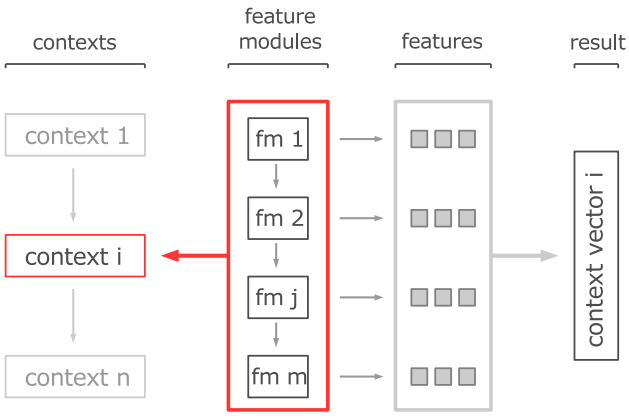


Abbildung 5.5: Analyse der kontextuellen Eigenschaften.

Definition 13 (Feature)**.** Sei ctx ein Kontext nach Definition 8 mit $ctx \in (\Sigma \backslash T)^*$. Ein **Feature** von ctx ist ein Tupel (wie definiert in [Grishin (2011)])

$$(id, f(ctx)) \qquad \text{mit } id \in \mathbb{N}, f : (\Sigma \backslash T)^* \to \mathbb{R} \tag{5.1}$$

bestehend aus einem ganzzahligen Identifikator id und einer reellwertigen Funktion f, die eine Eigenschaft des übergebenen Kontexts quantifiziert.

Definition 14 (Feature-Modul)**.** Sei ctx ein Kontext nach Definition 8 mit $ctx \in (\Sigma \backslash T)^*$. Ein **Feature-Modul** m ist eine Funktion, welche zu einem gegebenen Kontext $n = |m(ctx)|$ Features liefert:

$$m(ctx) = \{[i, f_i(ctx)]\}_{i=1}^{n} \qquad mit\ |m(ctx)| = n \tag{5.2}$$

Grundsätzlich kann zwischen *statistischen* und *heuristischen* Feature-Modulen unterschieden werden. Statistische Module führen ihre Analysen ohne *a-priori* Wissen über die Domäne aus, während heuristische Module häufig Gebrauch von domänenspezifischem Wissen machen. Nachfolgend sollen einige wichtige Vertreter beider Gruppen vorgestellt werden.

Statistische Feature-Module

Der Vorzug statistischer Feature-Module ist, dass sie kein zusätzliches Wissen über die Domäne, den Text-Corpus oder den Anwendungsfall benötigen. Sie eignen sich daher besonders für Klassifikations-Vorgänge, in denen sich diese Parameter häufig ändern. Die hier beschriebenen statistischen Feature-Module wurden insbesondere gewählt (wie beschrieben in Kapitel 2), um den orthographisch konsistenteren, häufig aus Stopworten bestehenden Kontext des zu klassifizierenden Terms auf wiederkehrende Muster zu untersuchen.

Kontext-Komposition Das Feature-Modul `ContextComposition` liefert die Menge aller Worte im zu untersuchenden Kontext, bildet somit ein Wort-Histogramm.

Die Reihenfolge der Worte im Kontext wird dabei nicht berücksichtigt.

Sei ctx ein Kontext (nach Definition 8) und wid eine Funktion, die jedem Wort (nach Definition 5) eines Corpus eine eindeutige Identifikationsnummer zuweist mit $wid : (\Sigma \backslash U)^* \rightarrow \mathbb{N}$. Weiterhin sei $f_{comp}(w, ctx)$ wie folgt auf einem Wort w und einem Kontext ctx definiert:

$$f_{comp}(w, ctx) = \begin{cases} 1, & \text{falls } w \text{ in ctx enthalten ist} \\ 0, & \text{sonst} \end{cases} \tag{5.3}$$

Das Feature-Modul m_{comp} zur Analyse der Kontext-Komposition ist dann die Menge aller Tupel, für die gilt:

$$m_{comp}(ctx) = \{[wid(w_i), f_{comp}(w_i, ctx)]\}_{i=1}^{|ctx|} \tag{5.4}$$

für alle Worte $w_i \in ctx$.

Co-Occurrence Das `Co-Occurrence` Feature-Modul untersucht paarweise Wort-Nachbarschaften im Kontext. Da die Abfolge der Worte grammatikalischen Regeln unterliegt, ist die Nachbarschaft zweier Worte nicht beliebig, sondern besitzt eine messbare Spezifität. Aufgrund der von diesem Modul gesammelten Statistiken sollen sich Vorhersagen ableiten lassen, mit welcher Wahrscheinlichkeit ein Wort der gesuchten Klasse auf diesen Kontext folgen wird.

Sei ctx ein Kontext (nach Definition 8) und $id(w, v)$ eine Funktion, die jedem Wort-Paar (nach Definition 5) eines Corpus eine eindeutige Identifikationsnummer zuweist mit $id : (\Sigma \backslash U)^* \times (\Sigma \backslash U)^* \rightarrow \mathbb{N}$. Dann ist $f_{coOc}(w, v, ctx)$ eine Funktion über einem Wort-Paar w, v und dem Kontext ctx wie nachfolgend definiert:

$$f_{coOc}(w, v, ctx) = \begin{cases} 1, & \text{falls } w, v \text{ in ctx benachbart sind} \\ 0, & \text{sonst} \end{cases} \tag{5.5}$$

Das Feature-Modul zur Analyse der Wort-Nachbarschaften ist somit die Menge aller Tupel, für die gilt:

$$m_{coOc}(ctx) = \{[id(w_i, w_{i+1}), f_{coOc}(w_i, w_{i+1}, ctx)]\}_{i=1}^{|ctx|-1} \tag{5.6}$$

für alle Worte $w_i \in ctx$.

Kontext-Komposition mit Distanz-Gewichtung Ähnlich wie das Co-Occurrence-Modul extrahiert das Modul `ContextCompositionDistance` die Abfolge der Worte im gegebenen Kontext, die aufgrund von Grammatik und Satzbau eine bestimmte Wahrscheinlichkeit besitzt. Dabei wird jedoch statt der Wort-Nachbarschaften die Position des Worts im Kontext ausgewertet. Dieser Idee liegt zugrunde, dass Worte, die dem zu klassifizierenden Wort näher sind, vermutlich eine höhere Signifikanz für den Klassifikationsvorgang besitzen als weiter entfernte.

Sei ctx ein Kontext (nach Definition 8) und $id(w, i)$ eine Funktion, die jedem Wort (nach Definition 5) eines Corpus an Position i im Kontext eine eindeutige Identifikationsnummer zuweist mit $wid : (\Sigma \backslash U)^* \times \mathbb{N} \to \mathbb{N}$. Dann ist $f_{compDist}(w, i, ctx)$ eine Funktion über einem Wort w, einer Positionsangabe i und dem Kontext ctx wie nachfolgend definiert:

$$f_{compDist}(w, i, ctx) = \begin{cases} \alpha \cdot i, & \text{falls } w \text{ in ctx an Position } i \text{ enthalten ist} \\ 0, & \text{sonst} \end{cases} \tag{5.7}$$

α ist dabei ein Gewichtungsfaktor, $\alpha = 1$ liefert beispielsweise ein mit der Position linear zunehmendes Gewicht.

Das Feature-Modul zur Analyse der distanz-gewichteten Kontext-Zusammensetzung ist somit die Menge aller Tupel, für die gilt:

$$m_{compDist}(ctx) = \{[id(w_i, i), f_{compDist}(w_i, i, ctx)]\}_{i=1}^{|ctx|} \tag{5.8}$$

für alle Worte $w_i \in ctx$.

Heuristische Feature-Module

Mit Hilfe von heuristischen Feature-Modulen lässt sich Wissen über die Domäne, welches a-priori vorhanden ist, in den Klassifikations-Ablauf integrieren (wie beschrieben in Abschnitt 5.1). Durch Auswertung von Nutzer-Feedback aus früheren Klassifikationsläufen können maschinelle Lernverfahren angestoßen werden, um zukünftige Klassifikationsläufe zu verbessern (wird in Abschnitten 5.1 und 5.3 besprochen). Sollte beispielsweise die Sprache des Text-Corpus bekannt sein, kann über eine *Part-of-Speech*-Analyse bei der Entschlüsselung der Semantik der Worte helfen. Weiterhin gibt es oft Datenbanken zu häufigen Fragestellungen, die als Heuristik verwendet werden könnten (häufige Ortsnamen-Bestandteile, Personenverzeichnisse, etc). Nachfolgend werden einige der verwendeten Heuristiken aufgezeigt.

Part-of-Speech Heuristik Part-of-Speech Analysen geben Aufschluss über die Funktion eines Wortes innerhalb eines Satzes (beispielsweise Substantiv, Artikel, etc). Im Gegensatz zur rein statistischen Auswertung der Abfolge der Worte lassen sich durch Verwendung einer Part-of-Speech-Heuristik sprachliche Muster erkennen, zum Beispiel, dass vor der gesuchten Klasse X häufig die Abfolge *Artikel, Adverb, Substantiv* zu finden ist.

Präfix- / Suffix-Heuristik Zu klassifizierende Worte können auf prominente Muster wie häufig verwendete Präfixe oder Suffixe untersucht werden. Beispielsweise sind bei Ortsnamen häufig Gemarkungsbeschreibungen wie „-furt“, „-bach“ oder „-berg“ als Suffix zu beobachten. Die Präfix- / Suffix-Heuristik kann einen Term und seinen Kontext auf das Vorhandensein solcher Indikatoren untersuchen.

Wörterbuch-Heuristik Falls domänenspezifisches Wissen in Form von Wörterbüchern für die Sprache des untersuchten Text-Corpus vorhanden ist, kann dieses nicht nur dazu verwendet werden, um Vorschläge des Klassifikations-Algorithmus

zu verifizieren, sondern auch, um initiale Gutmuster bereitzustellen. Existiert beispielsweise eine Liste häufiger männlicher Vornamen, kann diese verwendet werden, um einen Referenz-Vektor zu bilden, anhand dessen dann der Text auf weitere Instanzen untersucht wird.

Markup- / Struktur-Heuristik Wenn die untersuchte Textquelle hierarchisch strukturiert oder annotiert ist (beispielsweise *HTML*, *XML*, *JSON*, ...), kann die Struktur auf häufige Muster untersucht werden (beispielsweise: Instanzen der Klasse X „zumeist als Überschrift annotiert“ oder „häufig auf Hierarchie-Ebene z anzutreffen“. Das Ausnutzen struktureller Informationen zur Klassifikation wird beispielsweise in [Kluegl u. a. (2013)] angewendet.

In diesem Abschnitt wurden Feature-Module vorgestellt, mit deren Hilfe die Eigenschaften eines Kontexts serialisiert werden können. Nachfolgend wird nun der Vektorraum beschrieben, der durch die Gesamtheit der eingesetzten Feature-Module bestimmt wird.

5.2.4 Aufbau des Informationsraumes

Um Worte mit Hilfe des Vektorraum-Retrievals klassifizieren zu können, muss zunächst ein Vektorraum definiert werden. Danach können die kontextuellen Eigenschaften zweier Worte als Vektoren über diesem repräsentiert und anhand von Distanzfunktionen miteinander verglichen werden.

Das zuvor beschriebene klassische Beispiel des Vektorraum-Retrievals, relevante Dokumente zu einer Suchanfrage zu finden, wird für den vorliegenden Anwendungsfall verallgemeinert:

Anstatt den Vektorraum auf eine Menge von Schlagworten festzulegen, wird seine Dimensionalität durch die Anzahl der verfügbaren Feature-Module und zugehöriger Features definiert.

Für eine Menge $F = \{f_0, \ldots, f_{n-1}\}$ aus n Feature-Modulen ist die Dimensionszahl des Informationsraumes demnach die Summe alle Features aller Feature-Module:

$$dim = \sum_{i=0}^{n-1} |f_i|$$

Mit der Definition der Feature-Module (nach Definition 14) kann ein Kontext-Vektor nun wie folgt ausgedrückt werden:

Definition 15 (Kontext-Vektor)**.** Gegeben sei ein Term w mit k Auftreten innerhalb eines Texts. Seien dann $c_1, \ldots, c_k$ die resultierenden Kontexte von w_i, $1 \leq i \leq k$. Seien weiterhin $m_1, \ldots, m_m$ Feature-Module wie definiert in Definition 14. Dann ist der **Kontext-Vektor** $\vec{v}_w$ zu w eine Menge von Tupeln mit

$$\vec{v}_w = \bigcup_{j=1}^{m} \frac{1}{k} \sum_{i=1}^{k} m_j(c_k) \tag{5.9}$$

Abbildung 5.6 zeigt den schematischen Aufbau des resultierenden Kontext-Vektors.

Nach der Definition des Vektorraumes können nun Worte als Vektoren über diesem Vektorraum dargestellt werden. Mit Hilfe des in Algorithmus 2 vorgestellten Vorgehens kann dieses geleistet werden.

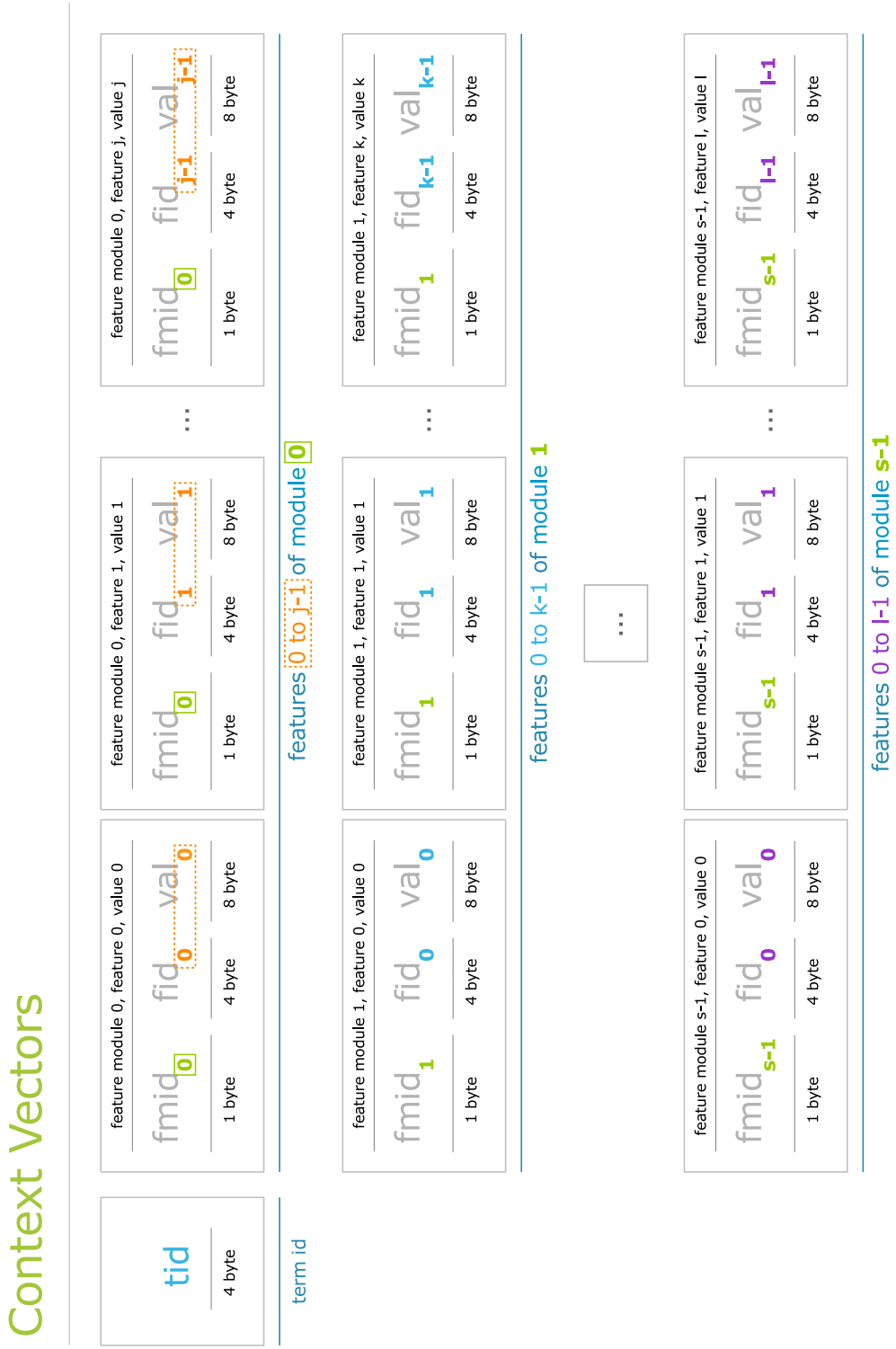


Abbildung 5.6: Schematischer Aufbau eines Kontext-Vektors mit s FeatureModulen nach Definition 15.

```
Data : Textcorpus C, Term t, Feature-Module F
Result : Kontext-Vektor v_t zu Term t
begin
    v_t ⟵ emptyVector();
    /* ermittelt alle Vorkommen von t in C                    */
    O ⟵ getOccurrences(t, C);
    /* für jedes Vorkommen o                                  */
    forall the o in O do
        /* Kontext c_o ermitteln                              */
        c_o ⟵ getContext(t, o);
        /* Kontext an alle Feature-Module übergeben           */
        forall the f in F do
            /* serialize context properties                   */
            props ⟵ analyzeContext(f, c_o);
            /* add to context vector                          */
            v_t ⟵ addToVector(props, v_t);
        end
    end
    return v_t;
end
```

Algorithmus 2 : Erzeugung eines Kontext-Vektors

Der Algorithmus erwartet als Eingabe einen Textcorpus C, auf dem die Analysen vollzogen werden sollen, einen Term t, für den eine Vektor-Darstellung gefunden werden soll und eine Menge von Feature-Modulen $F = \{f_0, \ldots, f_{n-1}\}$. Nach erfolgreichem Beenden liefert der Algorithmus einen Kontext-Vektor $\vec{v_t}$ für den Term t. Dazu werden zunächst alle Vorkommen O von t in C ermittelt. Für jedes Vorkommen $o \in O$ wird der jeweilige Kontext c_o extrahiert. Dieser Kontext wird nun jedem Feature-Modul aus F zur Analyse übergeben. Die Resultate dieser Analyse werden in einem Vektor komponentenweise addiert. Abbildung 5.6 zeigt den schematischen Aufbau des resultierenden Kontext-Vektors.

Vektoren, die auf diese Weise erzeugt werden, können nun unter Verwendung von Distanzmetriken verglichen werden.

5.2.5 Distanzmetriken

Um das Vektorraum-Retrieval als Klassifikator einsetzen zu können, bedarf es weiterhin der Definition eines Ähnlichkeitsmaßes. Zwei Vektoren a, b werden dann als zur gleichen Klasse gehörig betrachtet, wenn ihre *Ähnlichkeit* einen gewissen Wert θ übersteigt (oder ihr *Abstand* einen gewissen Wert unterschreitet):

$$sim(a, b) \geq \theta \Rightarrow class(a) = class(b)$$

oder

$$dist(a, b) \leq \theta \Rightarrow class(a) = class(b)$$

Nachdem der Vektorraum durch die Features der Feature-Module festgelegt ist, gilt es nun zu definieren, wie die Ähnlichkeit der Vektoren gemessen werden kann. Nachfolgend wird das bekannte *Cosinus-Maß* vorgestellt und seine Tauglichkeit für den vorliegenden Anwendungsfall untersucht.

Cosinus-Maß

In einem Vektorraum bietet sich zunächst die euklidische Distanz als Abstandsmaß zwischen zwei Punkten an. Ein prominentes, darauf basierendes Ähnlichkeitsmaß aus der Literatur (beispielsweise in [Baeza-Yates und Ribeiro-Neto (1999)]), welches auch für das eingangs genannte Beispiel, relevante Dokumente zu einer Suchanfrage zu finden, verwendet wird, ist das *Cosinus-Maß*.
Das Cosinus-Maß berechnet den Cosinus des Winkels zweier Vektoren a, b nach Gleichung 5.10.

$$cos(\theta) = \frac{a \cdot b}{||a||\ ||b||} \tag{5.10}$$

$||a||$ steht dabei für die Norm eines Vektors und wird wie in Gleichung 5.11 berechnet.

$$||a|| = \sqrt{\sum_{i=1}^{n} q_i^2} \tag{5.11}$$

Vorteile des Cosinus-Maßes Das Cosinus-Maß ist ein einfaches, schnell implementiertes und auf linearer Algebra basierendes Distanzmaß. $cos(\theta)$ kann Werte in $[0; 1]$ annehmen und macht den Begriff der Ähnlichkeit zweier Vektoren mit einer kontinuierlichen Größe greifbar. Weiterhin lassen sich die Komponenten der Vektoren über Gewichte modifizieren, um besonders wichtige Eigenschaften zu verstärken oder unwichtige abzuschwächen. Ein bekannter Vertreter dieser Gewichtungs-Strategie wird mit dem *tf-idf* Modell gestellt, welches in [Salton, Wong und Yang (1975)] beschrieben wird. Es drückt mit einem Gewicht unter Beachtung der relativen und totalen Häufigkeit des Terms in Dokument und Corpus aus, wie relevant ein Term innerhalb eines Corpus für ein gewisses Dokument ist.
Für den vorliegenden Anwendungsfall ist die Gewichtung der Vektor-Komponenten nach dem tf-idf-Modell jedoch ungeeignet, weil gerade die Stopworte, deren konsistente Schreibweise statistisch ausgewertet werden soll, eine niedrige Gewichtung aufgrund ihres häufigen Auftretens erhalten würden.

Nachteile Ein Faktor, der direkten Einfluss auf die Aussagekraft des Cosinus-Maßes hat, ist die Anzahl der Dimensionen des Vektorraumes: [Beyer u. a. (1999)] beschreibt den so genannten *Fluch der Dimensionalität*, der bei zunehmender Dimensionszahl des Vektorraumes den Unterschied zwischen der maximalen und der minimalen Distanz der Datensätze beliebig klein werden lässt.

Neuere Studien, zum Beispiel [Houle u. a. (2010)], belegen zwar, dass nicht die reine Dimensionszahl, sondern eher für den Klassifikationsvorgang irrelevante Dimensionen den Fluch der Dimensionalität verursachen, dennoch muss dieser Umstand bei

$$\lim_{d \to \inf} \frac{dist_{max} - dist_{min}}{dist_{min}} \to 0 \tag{5.12}$$

Abbildung 5.7: Fluch der Dimensionalität

der Komposition der für den Anwendungsfall gewählten Feature-Module berücksichtigt werden, da diese die Dimensionen des Vektorraumes definieren.

5.2.6 Klassifikation

Algorithmus 1 gibt vor, dass im Zuge des Klassifikations-Verfahrens jeder Kontext-Vektor $\vec{v_w}$ eines Terms w in einer Datenbank mit einem Referenz-Vektor $\vec{r}$ verglichen wird. Die Ähnlichkeit zweier Vektoren wird anhand der im vorherigen Abschnitt beschriebenen Distanz-Funktion gemessen. Bei einer ausreichend hohen Ähnlichkeit wird dann w als zur selben Klasse wie $\vec{r}$ zugehörig betrachtet. Abbildung 5.8 versucht das Konzept der Ähnlichkeit von Vektoren über dem von den Feature-Modulen aufgespannten Raum zu verdeutlichen. Intuitiv erkennt man zwei offensichtlich ähnliche Vektor-Darstellungen und eine stärker abweichende.

Auswahl der Referenz-Terme

Es ist naheliegend, dass die Wahl des Referenz-Terms großen Einfluss auf die Klassifikations-Qualität hat. Dabei spielen der verwendete Text-Corpus, die inhaltliche Ausrichtung der Quelle (Fachartikel, Sportzeitschrift, Internet-Suchphrasen) und die Spezifität des gewählten Begriffs (eindeutige Begriffe gegenüber mehrdeutigen) große Rollen. Beispielsweise ist es für den Klassifikations-Algorithmus ein schlechter Ausgangspunkt, anhand eines mehrdeutigen Referenz-Begriffs wie *„Essen“*, der innerhalb eines Textes als Toponym und als Bezeichner für Nahrung auftauchen kann, nach weiteren Toponymen zu suchen, da im resultierenden Vektor Eigenschaften beider Klassen erfasst wären. Gesucht ist demnach ein Vertreter der gesuchten Klasse, der möglichst repräsentativ ist.

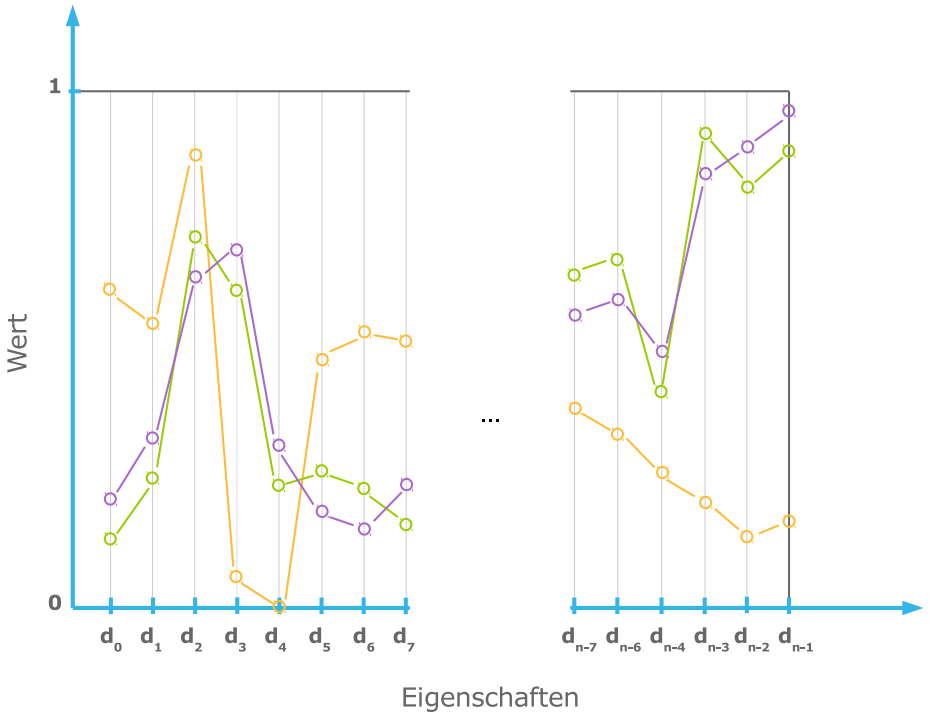


Abbildung 5.8: Schematische Visualisierung der Ähnlichkeit zwischen Kontext-Vektoren im Feature-Raum.

Werden weiterhin statt einem gleich mehrere Referenz-Terme als Beispiele der gesuchten Klasse übergeben, anhand derer eine Klassifikation vorgenommen werden soll, so gilt es, einen geeigneten Repräsentanten zu finden. In der Literatur wird dabei häufig der Begriff des *Centroids* oder des *geometrischen Zentrums* angeführt (beispielsweise in [Manning, Raghavan und Schütze (2008)]). Das geometrische Zentrum mehrerer Referenz-Vektoren erhält man durch Durchschnittsbildung:

Definition 16 (Referenz-Vektor)**.** Für $t_1, \ldots, t_n$ Referenz-Terme seien $\vec{v}_1, \ldots, \vec{v}_n$ die ihnen entsprechenden Kontext-Vektoren. Den resultierenden **Referenz-Vektor** $\vec{r}$ erhält man durch:

$$\vec{r} = \frac{1}{n} \sum_{i=1}^{n} norm(\vec{v}_i) \qquad norm(\vec{v}) = \frac{\vec{v}}{||\vec{v}||}$$

Nach der Wahl geeigneter Referenz-Terme kann eine Ähnlichkeitsmessung nach Algorithmus 1 und somit die Klassifikation erfolgen. Bevor Kapitel 8 die Leistungsfähigkeit dieses Ansatzes untersucht und dabei besonders auf die Eignung für die Domäne historischer Texte eingeht, stellt Kapitel 7 zunächst einige Details der Implementierung dieses Ansatzes vor. Insbesondere wurde in der Entwurfsphase berücksichtigt, dass der Ansatz in einer Verarbeitungskette zur automatisierten Metadaten-Extraktion eingesetzt werden soll.

5.3 Optimierung der kontextuellen Analyse

Der im letzten Abschnitt dieses Kapitels vorgestellte Ansatz greift den Kontext-Vektor des vorherigen Abschnitts auf und untersucht Möglichkeiten, die Klassifikations-Leistung zu optimieren.

5.3.1 Konzept

Ziel der in [Schneider (2013)] beschriebenen Arbeit ist die Optimierung der Klassifikations-Leistung des Kontext-Vektor Ansatzes. Dazu werden zunächst Strategien für die Wahl des Referenz-Vektors evaluiert, anhand dessen die Klassifikation nach Algorithmus 1 vorgenommen werden soll. Weiterhin wird untersucht, ob sich ein auf neuronale Netze stützendes Verfahren eignet, die für die Klassifikation relevanten Dimensionen des Kontext-Vektors zu lernen. Im weiteren Verlauf wird schließlich analysiert, ob eine Verbundsaussage der vorgestellten Ansätze für die Klassifikation von Worten in historischen Texten vorteilhaft ist.

Optimierung der Referenz-Menge

Die Auswahl eines geeigneten Vektors, anhand dessen die in Abschnitt 5.2.6 beschriebene Ähnlichkeitsmessung vorgenommen wird, hat große Auswirkungen auf die Qualität der Klassifikationsleistung. Es gilt daher, repräsentative Vertreter der

gesuchten Klasse zu finden.

In einem ersten Schritt wird zunächst untersucht, ob die Hinzunahme weiterer Referenz-Terme und anschließende Bildung des Durchschnittsvektors die Vorhersagequalität zu steigern vermag. Diese Variante zur Bildung des Referenz-Vektors hat allerdings den Nachteil, dass die resultierenden Vektoren durch die Durchschnittsbildung viele Dimensionen ansammeln können, die für die Klassifikationsqualität keine Signifikanz besitzen. [Beyer u. a. (1999)] nennt das den *Fluch der Dimensionalität*, weil durch diesen Effekt die Ähnlichkeit zweier Vektoren zueinander verringert wird.

In Kapitel 8 wird erwartungsgemäß belegt, dass durch die Hinzunahme weiterer Referenz-Terme die Klassifikationsqualität des Algorithmus 1 ab einem gewissen Punkt nicht gesteigert werden kann.

In einem weiteren Schritt wird untersucht, ob die Reduktion der Dimensionszahl des Referenz-Vektors zur Steigerung der Klassifikationsleistung beiträgt. Dazu eignet sich ein Modell, welches Dimensionen ein Gewicht zuordnet, welches ihrer Signifikanz für den Klassifikationsvorgang entspricht. Abschnitt 5.2.5 hat dazu schon das *tf-idf* Modell vorgestellt, welches die Relevanz eines Wortes für ein Dokument anhand der Frequenz seines Auftretens beziffert: Seltenen Worten wird dabei ein hohes, häufigen ein niedriges Gewicht zugewiesen. Für den vorliegenden Fall werden jedoch insbesondere Stopworte statistisch ausgewertet, da sie durch ihre Häufigkeit orthographisch konsistenter sind als selten verwendete Worte. Zur Ermittlung der relevanten Dimensionen eines Kontext-Vektors eignet sich demnach ein invertiertes Modell, welches häufige Terme mit einem hohen Gewicht versieht. In Kapitel 8 wird die Wirksamkeit dieser Maßnahme belegt.

Kontext-Vektor mit nachgelagertem Perzeptron

[Schneider (2013)] untersucht weiterhin alternative Wege, die Kontext-Vektoren zu klassifizieren: Da vom Nutzer für die Ausführung von Algorithmus 1 bereits Beispiele für die gesuchte Klasse erwartet werden, können diese verwendet werden, um ein künstliches neuronales Netz zu trainieren. Ein neuronales Netz ist laut [Fausett (1994)] ein informationsverarbeitendes System, welches die Funktionsweise von Nervensystemen in Lebewesen abzubilden versucht. Es basiert auf *Neuronen*, welche durch gewichtete Verbindungen untereinander verknüpft sind. Neuronen haben Ein- und Ausgabesignale, wobei das Ausgabesignal eines Neurons durch eine Aktivierungsfunktion bestimmt wird, die auf die Summe der Eingabesignale angewendet wird. Abbildung 5.9 stellt diese Funktionsweise schematisch dar.

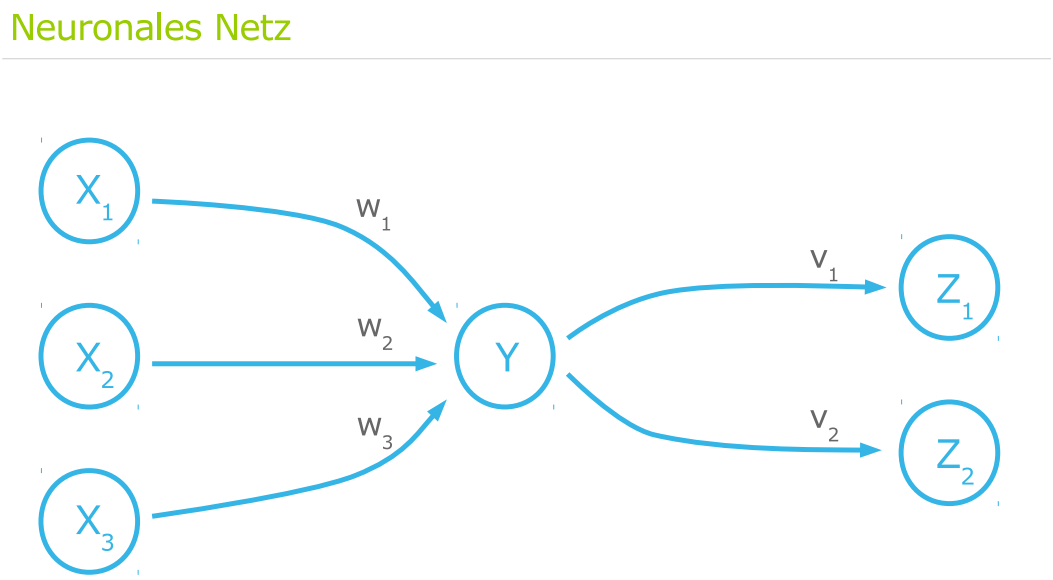


Abbildung 5.9: Simples neuronales Netz, schematische Darstellung.

[Rosenblatt (1958)] stellt mit dem Perzeptron ein einfaches neuronales Netz vor. Durch geeignete Lernverfahren (wie beschrieben in [Fausett (1994)] und genügend

viele Beispiele soll das so trainierte neuronale Netz in der Lage sein, auch unbekannte Eingaben zu klassifizieren.

Für den vorliegenden Anwendungsfall wird ein Perzeptron verwendet, welches so viele Eingabesignale besitzt wie es Dimensionen im Vektorraum der Kontext-Vektoren gibt (nach Definition 15). Das Ausgabesignal des Perzeptrons gibt an, ob der angelegte Kontext-Vektor der gesuchten Klasse angehört. Kapitel 8 informiert über die mit diesem Verfahren erzielte Klassifikationsleistung.

Verbundaussage durch Boosting

In Kapitel 8 wird ersichtlich, dass sowohl der Kontext-Vektor- als auch der Perzeptron-Ansatz Vorzüge bei der Klassifikation besitzen und in der Lage sind, mit der Domäne historischer Texte umzugehen.

In [Schneider (2013)] wird daher untersucht, ob sich Kombination der vorgestellten Klassifikatoren (Kontext-Vektor- und Perzeptron-Ansatz) durch Boosting zu einem Verbundsklassifikator positiv auf die Klassifikationsleistung auswirkt.

Definition 17 (Boosting)**.** Unter dem Begriff **Boosting** (engl. „Verstärken“) versteht man die gewichtete Kombination mehrerer Klassifikatoren zu einem Verbundsklassifikator. [Witten, Frank und Hall (2011)] kennt diesen Vorgang auch als *Bagging* und führt als einfachstes Beispiel die gemittelte Aussage aller beteiligten Klassifikatoren an, die in diesem Fall jeweils mit dem gleichen Gewicht an der Gesamtaussage beteiligt sind.
Die Gewichte der beteiligten Klassifikatoren sollen derart optimiert werden, dass der Vorhersagefehler des Verbundsklassifikators minimiert wird. In [Freund und Schapire (1997)] werden dafür geeignete Verfahren illustriert.

Der Definition folgend kombiniert der Klassifikator, der durch den Boosting-Ansatz gewonnen wird, den Kontext-Vektor-Ansatz und das Perzeptron zu einer gewichteten Verbundsaussage. In Kapitel 8 werden die qualitativen Verbesserungen der Klassifikatorleistung gegenüber den einzelnen Ansätzen dargestellt.

Dieses Kapitel hat verschiedene Verfahren demonstriert, die unter Verwendung der Information im Kontext eines Terms eine Klassifizierung vornehmen können. Vor der Leistungsbewertung dieser Verfahren in Kapitel 8 werden in den nachfolgenden Kapiteln noch einige Implementierungs-Details erläutert.

6 Implementierungs-Grundlagen

Bei der Implementierung der vorgestellten Konzepte wurde Wert auf Effizienz, Robustheit und Skalierbarkeit gelegt. Bevor in den nachfolgenden Kapiteln die Umsetzung beschrieben wird, sollen zunächst einige der dabei häufig verwendeten Begriffe und Konzepte definiert werden.

6.1 Framework

Zunächst gilt es den Begriff Framework zu erläutern: [Riehle (1998)] beschreibt ein Framework als eine Abstraktion, die eine generische, durch den Nutzer erweiterbare Programmfunktionalität zur Verfügung stellt. Der grundlegende Kontrollfluss im Programm wird dabei aber vom Framework diktiert und nicht vom Nutzer. Dieses Prinzip ist als das Entwurfsmuster *„Inversion of Control“* (IoC) bekannt, welches in [Fowler (2004)] näher dargestellt wird. Es ermöglicht den Entwicklern eine Anwendung zu entwerfen, welche eine Abfolge von Arbeitsschritten genau vorgibt, dem Nutzer aber die Wahl der Werkzeuge überlässt.
Ein prominentes Beispiel für die IoC-Architektur ist das *Servlet*-Modell aktueller Web-Anwendungen, in denen der Entwickler die Geschäftslogik innerhalb eines Servlets vorgibt, es aber dem Servlet-Container (dem Framework) überlässt, auf eine Http-Anfrage (Request) zu reagieren, den Aufruf an ein passendes Servlet zu übermitteln und den Rückgabewert (Response) der Geschäftslogik des Servlets an den Aufrufer weiterzureichen.

6.2 Aktoren-Modell

Das Aktoren-Modell (oder *actor model*) beschreibt ein Modell für nebenläufige Programme und wurde von [Baker und Hewitt (1977)] beschrieben. Einige Programmiersprachen (wie Scala[1] und Erlang[2]) besitzen Aktoren bereits als Sprachelemente, für andere Sprachen existieren Erweiterungen, welche die Funktionalität des Aktoren-Modells nachträglich bereitstellen (beispielsweise Akka[3] für *Java* - beschrieben im nachfolgenden Abschnitt, Theron[4] für *C++*).

Das Aktoren-Modell stellt Aktoren als nebenläufige Einheiten ohne geteilten Speicherbereich vor, die über asynchronen Nachrichtenaustausch miteinander kommunizieren. Jeder Aktor besitzt dabei eine eindeutige Adresse, einen Posteingang und ein Verhalten. An einen Aktor gesendete Nachrichten landen im Posteingang des Empfänger-Aktors, wo sie nach *first-in-first-out* Strategie einzeln verarbeitet werden. Als Reaktion auf eine Nachricht kann ein Aktor Nachrichten an andere Aktoren oder sich selbst schicken, neue Aktoren erzeugen und sein Verhalten ändern. Abbildung 6.1 illustriert das Konzept.

Da ein Aktor Nachrichten nur sukzessive bearbeiten kann und sein interner Zustand vor äußeren Zugriffen geschützt ist, entfällt die Notwendigkeit, den internen Zustand durch klassische Hilfsmittel aus der nebenläufigen Programmierung vor Veränderung zu schützen, beispielsweise durch wechselseitigen Ausschluss (Mutex), Semaphore und Signale. Durch die Abstraktion von diesen Nebenläufigkeits-Primitiven und die Ähnlichkeit zu Kommunikationsabläufen zwischen Menschen im alltäglichen Leben lassen sich nebenläufige Problemstellungen mit Hilfe des Aktoren-Modells intuitiv und präzise modellieren.

[1] http://www.scala-lang.org
[2] http://www.erlang.org
[3] http:/www.akka.io
[4] http://www.theron-library.com/

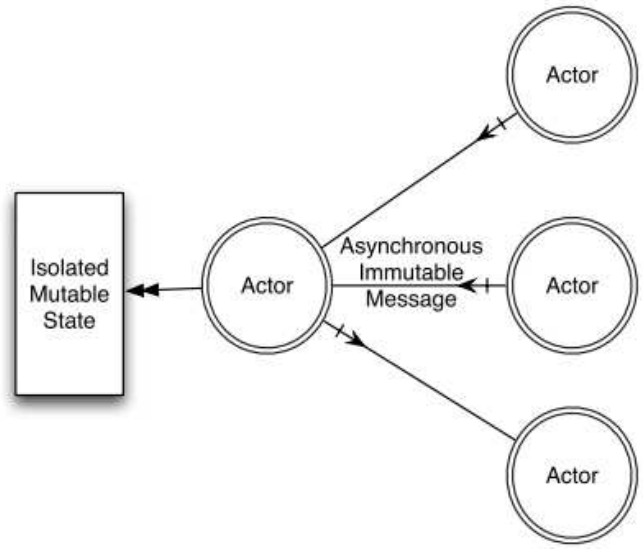


Abbildung 6.1: Schema Aktor-Modell. Quelle: [Club (2014)]

6.3 Aktor-Toolkit: Akka

Die in dieser Arbeit vorgestellten Konzepte wurden hauptsächlich in Java implementiert. Da in Java Aktoren kein grundlegender Bestandteil der Sprache sind, muss diese Funktionalität nachträglich zur Verfügung gestellt werden. *Akka* stellt eine Implementierung des Aktoren-Modells für die Java Virtual Machine bereit. Die Verwendung dieser Architektur bringt für die Umsetzung der bisher vorgestellten Konzepte folgende Vorteile mit sich:

Inhärente Nebenläufigkeit

Aktoren kommunizieren über asynchrone Nachrichtenverarbeitung - ähnlich zu den Kommunikationsabläufen zwischen Menschen. Nachdem ein Aktor (Sender) eine Nachricht an einen anderen Aktor (Empfänger) gesendet hat, kehrt der Aufruf sofort zurück und gibt dem Sender die Möglichkeit, eine andere Tätigkeit ausüben zu können, ohne bis zur vollständigen Bearbeitung der Nachricht durch den Empfänger warten zu müssen. Auch spielt es für den Sender keine Rolle, ob der Empfänger

die Nachricht selbst bearbeitet oder ob diese unter Umständen an einen anderen Aktor zur Bearbeitung delegiert wird.
Dadurch können nebenläufige, hoch-reaktive (da nicht blockierende) Architekturen erschaffen werden, die dynamisch auf Lastveränderungen reagieren können. In Abschnitt 7.4 auf Seite 83 wird dieses Konzept detailliert beschrieben.

Kompartmentalisierung

Ein zentrales Konzept eines robusten Entwurfs ist es, kritische Anwendungsbereiche zu isolieren und auf Fehler zu überwachen. Dieser Anforderung kann durch den Einsatz des Aktor-Modells des Akka-Frameworks Genüge getan werden.

Der Entwickler legt zunächst fest, welche Aktoren zur Lösung eines gegebenen Problems notwendig sind. Üblicherweise werden Aktoren definiert, welche die Arbeit koordinieren sowie Rückgabewerte und Änderungen im Lebenszyklus untergeordneter Aktoren überwachen. Weiterhin sind Aktoren notwendig, welche die (möglicherweise fehlschlagende) Arbeit ausführen. In einem System sind die verfügbaren Ressourcen limitiert (Threads, Hauptspeicher), weswegen Aktoren zu funktionalen Gruppen zusammengebunden werden, welche sich vorhandene Ressourcen teilen. Diese funktionale Zuordnung erfolgt über die Zuweisung mehrerer Aktoren zu einem gemeinsamen *Dispatcher*. Die Dispatcher verfügen über einen ihnen zugeordneten Threadpool und kontrollieren, welcher Aktor in welchem Thread welche Nachricht bearbeitet. Dazu wählt ein Dispatcher zunächst einen freien Thread, dann einen unbeschäftigten Aktor und eine Nachricht aus seiner Mailbox und lässt den Aktor die Nachricht bearbeiten. Nach der Bearbeitung wird der Aktor möglicherweise wieder aus dem aktiven Thread entfernt und muss auf die nächste Zuteilung durch seinen Dispatcher warten. Durch die Verwendung unterschiedlicher Dispatcher können Teile einer Anwendung voneinander abgeschottet und diese an die Konfiguration des Systems angepasst werden. Grafik 6.2 verdeutlicht dieses Prinzip.

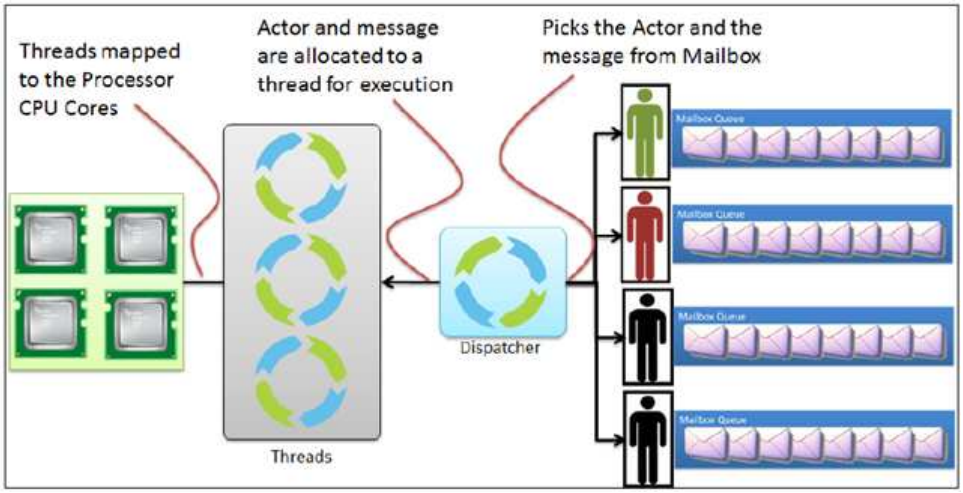


Abbildung 6.2: Schema Dispatcher. Quelle: [fxjwind (2014)]

Hierarchische Überwachung

Das Aktor-Modell, wie es vom Akka-Framework zur Verfügung gestellt wird, definiert Aktoren stets als Teil einer Hierarchie (siehe [http://akka.io (2013)]). Wie in Abbildung 6.3 zu sehen ist, ist Aktor C hierarchisch direkt den Aktoren D und E übergeordnet und ist damit ihr Eltern-Aktor. Die Aktor-Hierarchie ist dabei nicht auf eine einzige Virtuelle Maschine (VM) beschränkt, sondern kann mehrere lokale oder über ein Netzwerk-Protokoll verbundene VMs überspannen, wie im Beispiel an Aktor E zu sehen ist.

Ein Eltern-Aktor kann den Lebenszyklus seiner Kind-Aktoren überwachen: Sollte ein Aktor terminieren, wird dieses Ereignis an seinen direkten Eltern-Aktor propagiert, sodass dieser entsprechend darauf reagieren kann. Ein Eltern-Aktor hat unter anderem die Möglichkeit, Kind-Aktoren neu zu erzeugen. Dabei kann das Verhalten des Kind-Aktors so angepasst werden, dass es, sollte die Nachricht erneut auftreten, die zuvor zum Fehler führte, diese mit einer alternativen Bearbeitungsroutine erfolgreich bearbeiten kann. Für unterschiedliche Fehlertypen können damit verschiedene Herangehensweisen ausprobiert werden, bevor der Fehler als „nicht zu lösen“ an eine geeignete Stelle weitergereicht wird.

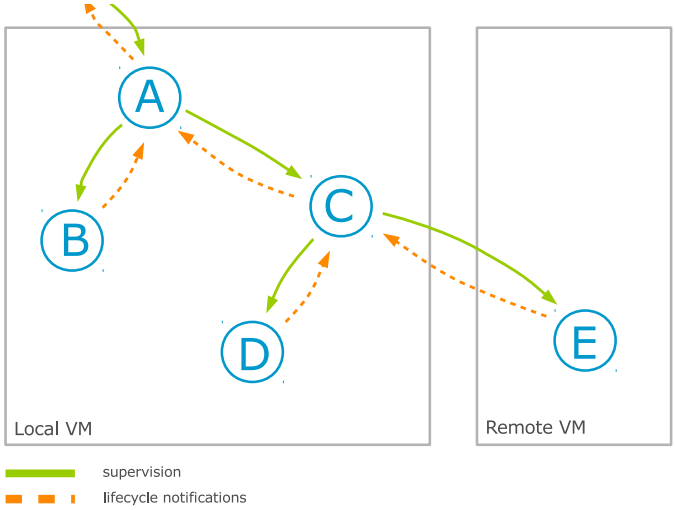


Abbildung 6.3: Schematische Darstellung einer Aktor-Hierarchie.

Kapitel 9.4 (Seite 120) beschreibt, wie mit Hilfe der hierarchischen Überwachung eine auch bei unerwartet auftretenden Fehlern robuste Anwendung geschaffen werden konnte.

Location Transparency

Wie zuvor beschrieben werden Aktoren stets als Teil einer Hierarchie arrangiert. Für zwei miteinander kommunizierende Aktoren spielt es aus Sicht der Programmlogik dabei keine Rolle, ob sich die Aktoren in der gleichen Virtuellen Maschine oder in zwei verschiedenen befinden. Dieser Effekt wird als *Transparenz des Ortes* (oder engl. *location transparency*) beschrieben [Kim und Agha (1995)].
Für den Entwickler ergibt sich dadurch der Vorteil, dass ganze Teilbäume der Aktoren-Hierarchie, welche innerhalb der Anwendung zum Einsatz kommt, in andere Virtuelle Maschinen ausgelagert werden kann, ohne dass dafür bestehender Programmcode modifiziert werden muss. Im Netzwerk miteinander kommunizierende Rechner können so leicht in die Bewältigung eines komplexen Problems mit eingebunden werden. Weiterhin kann unter Ausnutzung der örtlichen Transparenz die Geschäftslogik auf mehrere redundante Knoten ausgelagert und die Gefahr ei-

nes *Single Point of Failure*s (beschrieben in [Dooley (2001)]) reduziert werden.

Ein beliebtes Entwurfsmuster, welches im Zusammenhang mit der örtlichen Transparenz entstanden ist, stellt das *master-worker Pattern* [Wyatt (2013)] dar. Dieses Muster kommt in dieser Arbeit in einer erweiterten Form zum Einsatz und erlaubt es, im Netzwerk verbundene Rechner fehlertolerant zur Bewältigung komplexer Arbeitsschritte hinzuzuziehen, ohne diese im Vorfeld kennen zu müssen. Das Muster wird in Abschnitt 9.4 genauer beschrieben und seine Anwendbarkeit auf die vorliegenden Problemstellungen untersucht.

6.4 Monaden

Definition 18 (Monade)**.** Eine Monade `M<T>` über dem zugrunde liegenden Typ `T` hat folgende Komponenten:

1. Einen Typkonstruktor, der einen zugrunde liegenden Typ in den korrespondierenden Monadentyp überführen kann
2. Eine Einheitsfunktion `return`, die einen Wert des zugrunde liegenden Typs in den korrespondierenden Monadentyp überführen kann
3. Einen Binde-Operator `>>=` vom Typ `(M<T>)` $\rightarrow$ `(T`$\rightarrow$ `M<U>)` $\rightarrow$ `(M<U>)`. Der erste Parameter ist ein Wert im Monadentyp, der zweite Parameter ist eine Funktion, die Werte vom zugrunde liegenden Typ des ersten Parameters auf einen zweiten Monadentyp abbildet, und das Resultat ist vom zweiten Monadentyp.

Dabei müssen die **monadischen Gesetze** eingehalten werden:

1. *Assoziativität* des Binde-Operators:
   `(m >>= f) >>= g` $\equiv$ `m >>= (\x` $\rightarrow$ `(f x >>= g))`
2. *Neutralität* der Einheitsfunktion
   - `(return x) >>= f` $\equiv$ `f x`

- `m >>= return` $\equiv$ `m`

Wie in [Jones und Duponcheel (1993)] beschrieben, sind Monaden Strukturen aus der Welt der *Funktionalen Programmierung*, die einen Basistyp um Funktionalität erweitern und Berechnungen als eine Abfolge von Schritten darstellen können. Monaden erlauben es damit, abstrakte Pipelines von Arbeitsschritten zu definieren, deren konkrete Umsetzung zur Compile-Zeit noch nicht feststehen kann. Die bald (Stand März 2014) erscheinende Version 8 wird zusätzliche monadische Konstrukte wie Streams in die Sprache Java einführen [Oracle (2014)].

Container wie Listen und Multimengen sind Monaden, wenn eine an sie übergebene Funktion auf die enthaltenen Elemente angewendet wird und die Resultate vereinigt werden.

Ein weiteres Beispiel dafür sind Akkas monadische Erweiterung der *Futures*, welche nachfolgend vorgestellt werden soll.

6.5 Futures

Das Konzept der Futures wird beispielsweise in [Baker und Hewitt (1977)] erläutert und beschreibt ein Platzhalter-Objekt für eine Aktion, die nebenläufig ausgeführt wird und deren Ergebnis zu einem unbestimmten, späteren Zeitpunkt erfragt werden kann. Akkas Future-Implementierung geht über diese Funktionalität noch hinaus, indem sie einige monadische Methoden definiert [Typesafe (2013)] und [Haller u. a. (2013)].

Die monadische Eigenschaft der Future-Implementierung von Akka erlaubt eine Komposition von zwei Futures *Future<S> s, Future<T> t* derart, dass das Ergebnis des ersten Futures s als Eingabe an das zweite Future t übergeben werden kann, obwohl es erst zu einem unbestimmten Zeitpunkt in der Zukunft verfügbar ist. Auf diese Weise können abstrakte und nebenläufige Arbeitsschritte miteinander verkettet werden, obwohl ihre konkrete Umsetzung zur Compile-Zeit noch

nicht feststeht. Listing 6.1 verdeutlicht dieses Konzept: Zu sehen ist, wie mehrere nebenläufige Aktionen definiert werden, die bei Abschluss den nächsten, ebenfalls asynchronen Vorgang in einer Pipeline auslösen.

```
0  // -- create indexes
1
2  // sentence index
3  final Future<Boolean> fSentence = createSentenceIndex(s, source, sOut,
4      ec);
5
6  // word index
7  final Future<Boolean> fWord = createWordIndex(p, source, wOut, ec);
8
9  // (wait for completion)
10 final List<Future<Boolean>> indexFutures = new ArrayList<>();
11 indexFutures.add(fSentence);
12 indexFutures.add(fWord);
13 final Future<Iterable<Boolean>> indexFuturesCompleted = sequence(
14     indexFutures, ec);
15
16 // inverted index
17 final Future<Boolean> result = createInvertedIndex(
18     indexFuturesCompleted, sOut, wOut, i, iOut, p, ec);
19
20 // create vector index and shutdown
21 final Future<Boolean> finalResult = createVectorDB(result, label, c,
22     cOut, ec).andThen(shutdown(ec), ec);
23
24 return Await.result(finalResult, TIMEOUT);
```

Listing 6.1: Monadische Komposition mehrerer Futures

In Abschnitt 7.4 auf Seite (83) wird erläutert, wie die monadischen Future-Erweiterungen dazu verwendet werden können, eine dynamische Pipeline aus asynchronen, abstrakten Arbeitsschritten zu erstellen.

7 Implementierung: Klassifikations-Framework

Dieses Kapitel beschreibt die Entwicklung eines Frameworks zur Klassifikation von Termen in Texten, welches auf die in Kapitel 5 vorgestellten Ansätze zurückgreift. Die beschriebenen Techniken eignen sich besonders für den Einsatz in syntaktisch schwierigen, häufig wechselnden Domänen. In den nachfolgenden Abschnitten werden neben der allgemeinen Funktionsweise und Bedienung des Frameworks die verwendeten Datenstrukturen beschrieben sowie wichtige Implementierungs-Details beleuchtet. Das Kapitel schließt mit Anwendungsbeispielen sowie einer Bewertung der Resultate ab.

7.1 Konzept

Das vorgestellte Klassifikations-Framework definiert analog zu dem in den Grundlagen definierten Begriff *Framework* (Abschnitt 6.1 auf Seite 63) die grundlegende Abfolge der Arbeitsschritte eines abstrakten Klassifikations-Workflows, bei dem die Konkretisierung durch den Anwender erfolgen muss, der in Abhängigkeit von der untersuchten Domäne geeignete und für den jeweiligen Anwendungsfall spezialisierte Klassen zur Umsetzung der Arbeitsschritte wählt. Abbildung 7.1 zeigt eine schematische Darstellung der Schritte im Klassifikations-Workflow. Bei der Implementierung wurde neben Effizienz besonderes Augenmerk auf eine flexible Parametrisierung des Klassifikationsvorgangs sowie eine leichte Erweiterbarkeit gelegt.

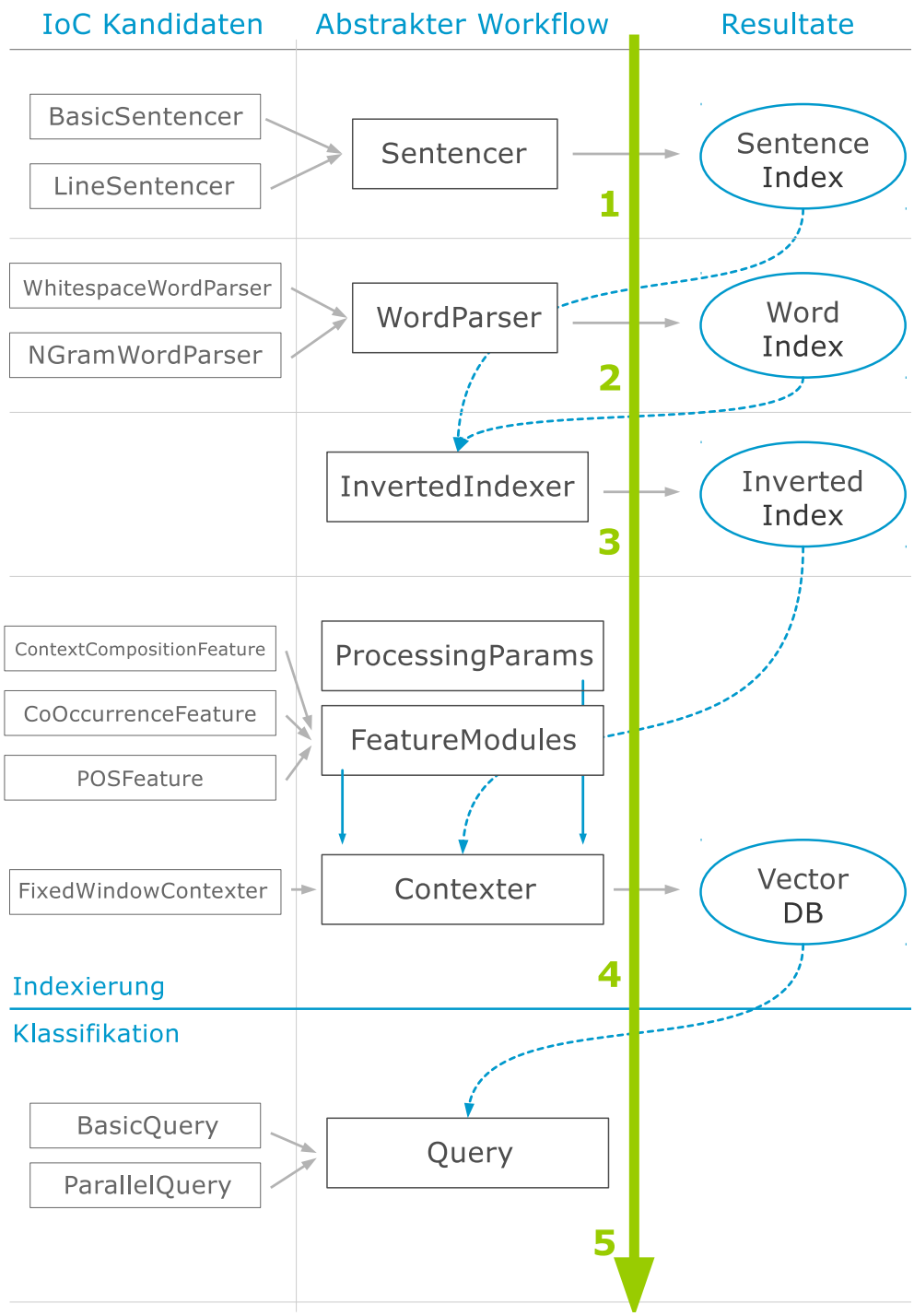


Abbildung 7.1: Schematische Darstellung des Klassifikationsworkflows.

Als Eingabe werden vom Anwender der zu untersuchende Text-Corpus sowie eine Klassifikations-Konfiguration erwartet, die auf die Corpus-Domäne zugeschnitten ist. Abschnitt 7.4 zeigt den Aufbau und Verwendung einer solchen Konfiguration. Der übergebene Corpus wird nun zunächst in ein indexiertes Format überführt, um einen effizienten Klassifikations-Vorgang zu ermöglichen (siehe Abschnitt 7.3 für eine detaillierte Erläuterung der Index-Erstellung). Mit Hilfe der indexierten Strukturen und der Klassifikations-Konfiguration kann eine Vektor-Datenbank erstellt werden, die eine Vektor-Repräsentation der kontextuellen Eigenschaften jedes Terms des Corpus enthält. In Abschnitt 7.3 wird dieser Vorgang ausführlich beschrieben. Die Index-Strukturen und Vektor-Datenbank lassen sich nun verwenden, um Klassifikations-Anfragen zu dem gegebenen Text-Corpus stellen. Die Ausgabe des Klassifikations-Frameworks ist eine Liste von Termen, von denen angenommen wird, dass sie die gleiche Klassifikation besitzen wie die Terme der Anfrage. Beispielhafte Anfragen und ihre Auswertung werden in Abschnitt 7.4 vorgestellt.

7.2 Zielsetzung

Ziel ist die Entwicklung eines Verfahrens, welches alle Terme eines gegebenen Text-Corpus als Vektoren über einem Eigenschaftsraum repräsentieren kann (wie beschrieben in Kapitel 5 auf Seite 38) und eine Suche nach ähnlichen Vektoren ermöglicht. Die Dimensionen des Eigenschaftsraumes werden durch Module bestimmt, die vom Nutzer in Abhängigkeit von dem zu untersuchenden Dokument und dessen Domäne anzupassen sind.

Als Ausgabe werden alle Terme eines Text-Corpus erwartet, deren Ähnlichkeit zu einem Suchvektor einen Schwellwert überschreiten.

Bei der Zielsetzung wird berücksichtigt, dass dieses Verfahren als Baustein in einer Verarbeitungskette für historische Metadaten eingesetzt werden soll. Da es hierbei häufig zu Anpassungen für das jeweils untersuchte Quell-Dokument und dessen

Verwendungszweck kommen kann, müssen die Verarbeitungsschritte flexibel angepasst werden können. Das Verfahren wurde daher nach dem allgemeinen Konzept eines Frameworks entworfen, bei dem die abstrakten Arbeitsschritte zwar vorgegeben, ihre konkrete Umsetzungen allerdings erst zur Laufzeit bekannt sind. Zur Steigerung der Effizienz sollen unabhängige Arbeitsschritte asynchron und parallel ausgeführt werden können.

7.3 Klassifikations-Workflow

Im nachfolgenden Teil werden die abstrakten Arbeitsschritte beschrieben, die innerhalb des Klassifikations-Workflows anfallen und die dabei erzeugten Datenstrukturen erläutert. Abbildung 7.1 zeigt die schematische Abfolge der einzelnen Schritte und die Beziehungen der erzeugten Datenstrukturen untereinander, sowie Beispiele für Klassen, die konkrete Umsetzungen der abstrakten Arbeitsschritte darstellen.

Schritt 1: Erstellung des Satz-Indexes

Zur effizienten Verarbeitung muss das Quelldokument in eine indexierte Struktur überführt werden. Der erste dafür notwendige Schritt (sichtbar auch in Abbildung 7.1, Schritt 1) wird vom `Sentencer` vorgenommen. Diese Schnittstelle übernimmt die Zerlegung des Ausgangstexts in Sätze (wie definiert auf Seite 24). Die Regeln, nach denen das Dokument in Sätze zerlegt wird, können in Abhängigkeit vom Dokument und dessen Domäne stark variieren. Es obliegt dem Nutzer, eine geeignete Implementierung dieser Schnittstelle zu wählen. Unter Verwendung des eingangs erklärten Entwurfsmusters *Inversion of Control* wird dann die vom Nutzer gewählte Implementierung im abstrakten Klassifikations-Workflow eingesetzt. Abbildung 7.1 zeigt beispielhafte konkrete Kandidaten für abstrakte Arbeitsschritte in der Spalte „IoC Candidates". Zwei mögliche Umsetzungen des

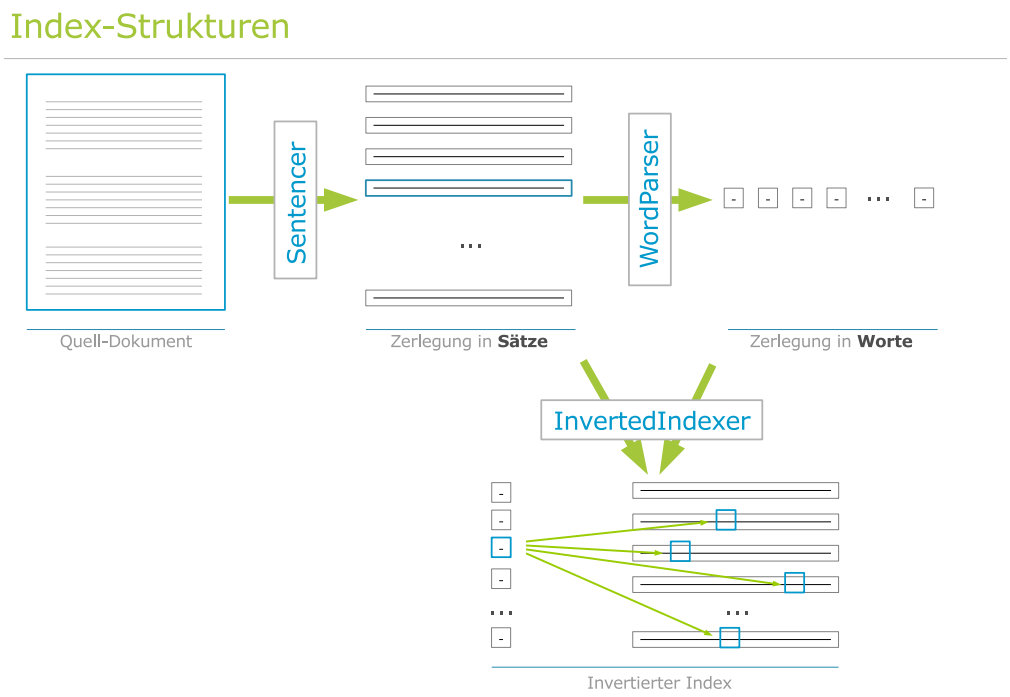


Abbildung 7.2: Schematische Darstellung der Datenstrukturen.

Sentencers wären beispielsweise, jede Zeile des Quell-Dokuments als Satz zu betrachten (Klasse `LineSentencer`) oder das Dokument an Punkten in Sätze zu trennen (Klasse `BasicSentencer`). Die Sätze werden nach der Zerlegung mit einer einzigartigen Identifikationsnummer versehen und im Satz-Index gespeichert. Dieser besteht aus einem Kopf- und einem Rumpfteil und ermöglicht es, einen Satz anhand seiner Identifikationsnummer aus dem Index zu erfragen. Das Konzept dieser Index-Struktur wird in Abschnitt 7.4 genauer erläutert und in Abbildung 7.3 anschaulich gemacht.

Schritt 2: Erstellung des Term-Indexes

Der zweite Arbeitsschritt (vgl. Abbildung 7.1, Schritt 2), den das Klassifikations-Framework auf dem Weg zur indexierten Struktur vorgibt, ist die Zerlegung der

im vorherigen Schritt vom Sentencer erzeugten Sätze in Terme (wie definiert auf Seite 24). Die Aufgabe der Schnittstelle `WordParser` ist es demnach, jeden vom Sentencer erzeugten Satz in Terme aufzubrechen. Programmausdruck 7.1 zeigt die Schnittstelle. Wie zuvor beim Sentencer sind auch beim WordParser verschiedene Varianten (und damit verschiedene IoC-Kandidaten) denkbar, nach welchen Regeln ein gegebener Satz in Terme zerlegt werden soll: Zwei Beispiele wären eine Trennung an Leerzeichen (Klasse `WhitespaceWordParser`) oder eine n-Gramme erzeugende Implementierung (Klasse `NGramWordParser`). Ganz analog zum Satz-Index werden die vom WordParser erzeugten Terme in einem Term-Index hinterlegt und in zuvor beschriebener Weise gespeichert (wie in Abschnitt 7.4 erläutert).

```
0  /**
1   * Splits a given sentence into tokens.
2   *
3   * @param sentence
4   *            The sentence to split.
5   * @return An array of String-tokens or null, if the sentence was null or
6   *         empty.
7   */
8  public abstract String[] tokenize(final String sentence);
```

Listing 7.1: Schnittstelle 'WordParser'

Schritt 3: Erstellung des Invertierten Indexes

Sind Satz- und Term-Index erzeugt, kann mit Hilfe dieser Datenstrukturen im dritten Arbeitsschritt des Klassifikations-Workflows nun ein dritter Index erzeugt werden. In diesem invertierten Term-Index (in Anlehnung an [Baeza-Yates und Ribeiro-Neto (1999)]) wird zusätzlich hinterlegt, in welchen (in Schritt 1 vom Sentencer erzeugten) Sätzen jeder (vom WordParser in Schritt 2 erzeugte) Term des Quelldokuments auftaucht. Dieser Arbeitsschritt ist in Abbildung 7.1 als Schritt 3 zu sehen. Die Klasse `InvertedIndexer` erzeugt Trefferlisten, die angeben, in welchen Sätzen im Satz-Index ein gegebener Term zu finden ist. Wie zuvor wird auch der invertierte Index in einer Kopf- und Rumpf-Struktur abgespeichert, mit dem Unterschied, dass der n-te ganzzahlige Eintrag im Kopfteil nun einen Wert enthält, der codiert, in welcher Entfernung zum Anfang des Indexes die Trefferliste

für den n-ten Term aus dem Term-Index beginnt (vgl. Abschnitt 7.4).

Schritt 4: Erstellung der Vektor-Datenbank

Unter Verwendung der bisher erzeugen Index-Strukturen kann jetzt das Quelldokument effizient durchsucht werden. Es gilt nun, die kontextuellen Eigenschaften aller Terme des Quelldokuments zu analysieren. Dies entspricht Arbeitsschritt 4 im abstrakten Klassifikations-Workflow, wie in Abbildung 7.1 dargestellt.

Die Schnittstelle `Contexter` hat die Aufgabe, den Kontext (wie definiert auf Seite 25) zu einem gegebenen Term zu extrahieren. Wie zuvor auch schon beim Sentencer und WordParser gibt es unterschiedliche Implementierungen, die in Abhängigkeit vom Quelldokument und dessen Domäne gewählt werden können. Eine mögliche Implementierung ist ein Contexter, der eine feste Anzahl von Termen rund um den zu untersuchenden Term als Kontext definiert (Klasse `FixedWindowContexter`). Nachdem der Kontext eines Terms in geeigneter Weise extrahiert wurde, gilt es dessen Eigenschaften zu analysieren und serialisieren. Diese Aufgabe wird von der Schnittstelle `FeatureModule` übernommen. Der Anwender kann dem Contexter eine Liste von FeatureModulen zuweisen, deren Analyse-Resultate dann im Kontext-Vektor gespeichert werden. In Kapitel 5 wurde dieser Vorgang ausführlich in der Theorie erläutert, die nachfolgenden Abschnitte gehen nun auf einige Implementierungsdetails ein.

Innerhalb eines Kontext-Vektors werden die kontextuellen Eigenschaften eines Terms gespeichert, wie sie im gesamten untersuchten Text-Corpus auftauchen. Dazu extrahiert der Contexter zunächst den Kontext zu einem gegebenen Term und reicht diesen dann an eine Liste von `FeatureModule`n weiter, welche dann die Eigenschaften des Kontexts serialisieren. Ein FeatureModul untersucht den übergebenen Kontext auf eine bestimmte Eigenschaft (wie beschrieben in Abschnitt 5.2.3 auf Seite 44) und muss dazu lediglich eine schmale Schnittstelle erfüllen (zu sehen in Programmausdruck 7.2). Jedes FeatureModul hat eine einzigartige Identifikations-Nummer und weist jeder Eigenschaft, die während der Analyse

gefunden wird, eine innerhalb des FeatureModuls einzigartige Identifikationsnummer zu. Durch die Kombination der FeatureModul-Identifikationsnummer mit der Feature-Identifikationsnummer entsteht eine global eindeutige Identifikation für jede Komponente des Kontext-Vektors. Grafik 5.6 zeigt den schematischen Aufbau eines Kontext-Vektors. Die Vektoren werden nun ebenfalls in einem Header-Body-Index gespeichert (beschrieben in Abschnitt 7.4), so dass man effizient auf den Kontext-Vektor für einen gegebenen Term zugreifen kann.

```
0  /**
1   * analyze the given context
2   *
3   * @param context
4   * @return FeatureCategory
5   */
6  public abstract FeatureCategory analyzeContext(String[] context);
```

Listing 7.2: Schnittstelle 'FeatureModule'

7.4 Implementierung

Da die Kontext-Analyse ein rechenintensiver und zeitaufwändiger Prozess ist, muss bei der Implementierung neben leichter Anpassbarkeit an wechselnde Domänen und Klassifikations-Umgebungen besonderer Wert auf die Effizienz des Klassifikations-Frameworks gelegt werden, um einen Einsatz in Echtzeit als Expertensystem oder Vorschlagswesen zu ermöglichen. Es gilt daher, geeignete Strukturen für die Speicherung der indexierten Daten zu erzeugen und diese möglichst effizient auszuwerten. Die nachfolgenden Abschnitte demonstrieren die in diesem Framework verwendeten Header-Body-Indexstrukturen und zeigen, welche Schritte zur Steigerung der Abfrage-Effizienz vorgenommen wurden.

Header-Body-Indexstruktur

Die Header-Body-Indexstruktur setzt den Framework-Charakter auch auf der Ebene der Datenstrukturen fort: Der abstrakte Klassifikations-Workflow schreibt eine Erstellung von indexierten Datenstrukturen vor, jedoch kann zur Entwurfszeit noch nicht bekannt sein, welche Implementierungen der Anwender zur Umsetzung

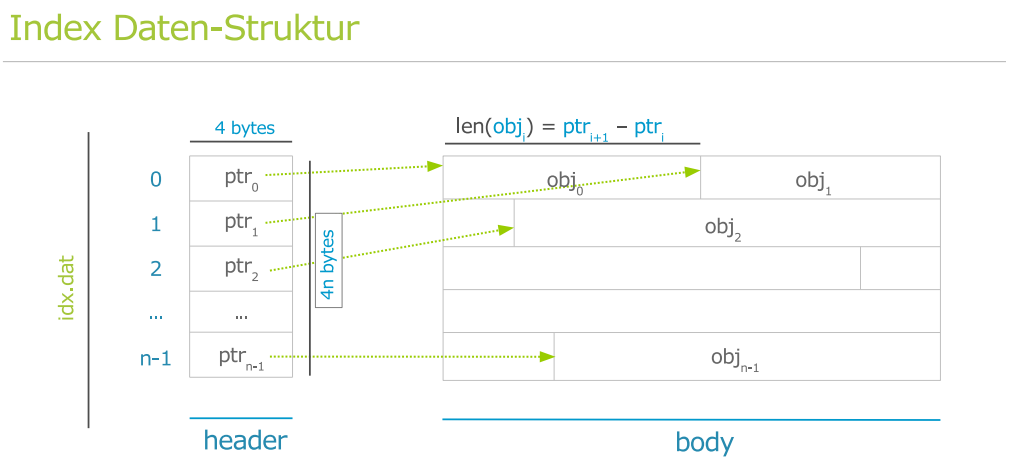


Abbildung 7.3: Schematische Darstellung der Header-Body-Indexstruktur.

der einzelnen Arbeitsschritte wählt und daher ebenfalls nicht, welcher Art die Objekte sind, die im Index gespeichert werden sollen.

Die Header-Body-Indexstruktur beschreibt eine allgemeine Datenstruktur, um auf Objekte unbekannter Länge mit einer eindeutigen Identifikationsnummer zugreifen zu können und besteht aus Kopfteil und Rumpfteil. Der Kopfteil besteht aus Ganzzahl-Werten, welche derart mit dem Rumpfteil verknüpft sind, dass der n-te ganzzahlige Wert im Kopfteil angibt, in welcher Entfernung zum Anfang des Indexes sich das n-te Objekt (das Objekt mit der Identifikationsnummer n) befindet. Die Länge des n-ten Objekts erhält man durch Differenzbildung zum Versatz-Wert des $n+1$-ten ganzzahligen Wert im Kopfteil. Grafik 7.3 illustriert dieses Konzept. Die Klasse `ByteBufferIterator` wird verwendet, um die Objekte wieder aus dem Index auszulesen (siehe Programmausdruck 7.3). Es ist dem Framework damit möglich, eine indexierte Datenstruktur zu erstellen und zu verwenden, ohne die konkrete Ausprägung der gespeicherten Objekte zu kennen.

```
0  public class ByteBufferIterator extends AbstractByteBufferIterator<ByteBuffer>
1  {
2    @Override
3    public ByteBuffer next()
4    {
5      if (!hasNext())
6        throw new NoSuchElementException("ByteBufferIterator#next()");
7
8      // read body data
9      byte[] data = new byte[(next - current)];
10     buff.position(current);
11     buff.get(data, 0, next - current);
12     buff.reset();
13
14     // calc next header pointers
15     nextHeaderPointers();
16
17     return MappedByteBuffer.wrap(data);
18   }
19 }
```

Listing 7.3: Klasse 'ByteBufferIterator' extrahiert Objekte aus Header-Body-Index

Konfigurations-Klassen

Da zur Compile-Zeit noch nicht feststehen kann, auf welche Domäne die Klassifikation angewendet wird, wird die Komposition der an dem Prozess beteiligten Klassen in die Laufzeit verschoben. Abbildung 7.4 zeigt ein zentrales Konfigurations-Objekt, welches Referenzen auf alle Schnittstellen enthält, die für die Erzeugung der domänenspezifischen Daten- und Abfragestrukturen notwendig sind. Es obliegt nun dem Nutzer, je nach Anwendungsfall konkrete, die Schnittstellen implementierende Objekte in einem spezifischen Konfigurationsobjekt zusammenzufassen.

Dazu wird Gebrauch vom „Inversion of Control“ Muster (IoC) gemacht, wie es in [Fowler (2004)] beschrieben wird. Dieses Muster erlaubt es, mit Hilfe eines Erzeuger-Objekts zur Laufzeit dynamisch ein auf einen Anwendungsfall zugeschnittenes Objekt-Netz zu erstellen. Insbesondere wird hierbei auf die Technik der „Dependency Injection“ zurückgegriffen, die vom Spring-Framework[1] in Form von speziell annotierten Konfigurations-Klassen [springsource.com (2013)] zur Verfügung gestellt wird. Programmausdruck 7.4 zeigt an einem Beispiel, wie ein solches Konfigurationsobjekt aussehen kann. Zu sehen ist, wie sich der Anwender

[1] http://www.springsource.org/

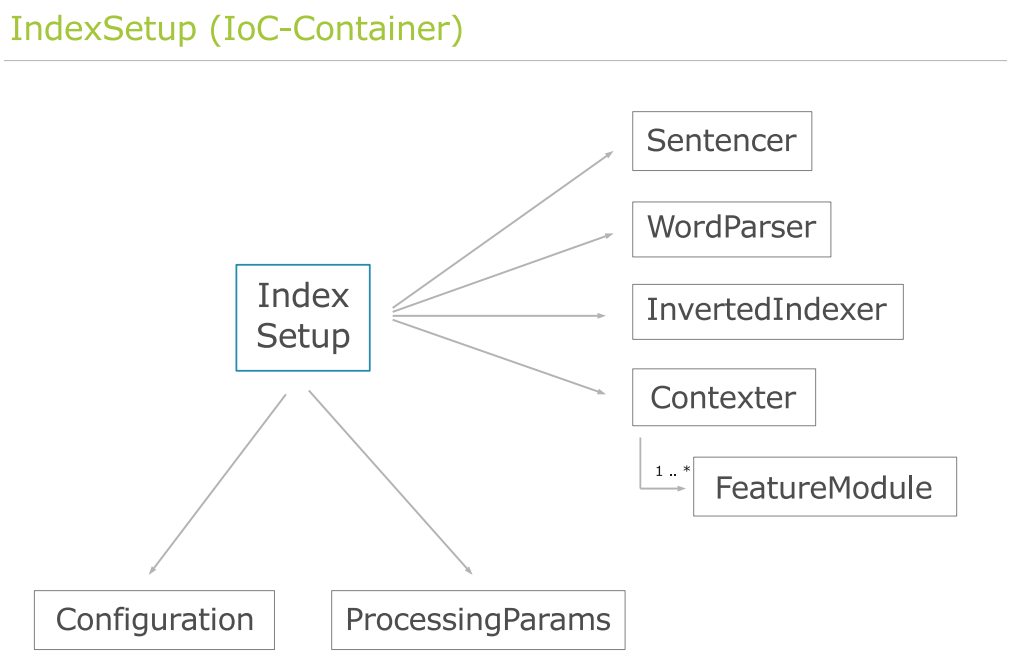


Abbildung 7.4: Komposition des IoC-Containers.

für eine bestimmte Ausprägung der am Workflow beteiligten Schnittstellen entschieden hat (Zeilen 12, 26 und 29). Für verschiedene Domänen lassen sich damit schnell angepasste Konfigurationen erstellen, ohne dass dadurch der zugrundeliegende Klassifikations-Workflow verändert werden muss.

```
0  @Configuration
1  public class BasicConfig extends AbstractConfiguration {
2    private final static int WINDOWSIZE = 5;
3
4    public BasicConfig() throws IOException {
5      super();
6    }
7
8    @Bean
9    @Override
10   public Contexter contexter() {
11     final Contexter c = new LeftWindowContexter(WINDOWSIZE);
12
13     // add feature modules
14     for (FeatureModule fmod : featureModules())
15       c.addFeatureModule(fmod);
16
17     return c;
18   }
19
20   @Override
21   public Set<FeatureModule> featureModules() {
22     final Set<FeatureModule> modules = new HashSet<>();
23
24     // statistical module: context composition
25     modules.add(new ContextCompositionFeature());
26
27     // statistical module: context composition, boosted by distance
28     modules.add(new ContextCompositionDistanceFeature(WINDOWSIZE));
29
30     return modules;
31   }
32 }
```

Listing 7.4: IoC Beispiel: Basis-Konfiguration

Optimierung der Workflow-Effizienz

Der bisher beschriebene Klassifikations-Workflow ist ein zeit- und rechenintensiver Prozess. Um den Einsatz innerhalb eines Experten-Systems oder einer anderen Echtzeit-Anwendung zu ermöglichen, gilt es nun, zeitaufwändige Programmabschnitte zu identifizieren und die Zeit zu reduzieren, die für die Beantwortung einer Anfrage notwendig ist.

Bei einer Klassifikations-Abfrage wird jeder Vektor innerhalb der Vektor-Datenbank mit dem Referenz-Vektor verglichen und bei hoher Ähnlichkeit in die Ergebnismenge aufgenommen (siehe Kapitel 5.2.5 zu den Distanzmetriken). Eine Partitionierung der Vektor-Datenbank und parallele Berechnung der Ähnlichkeiten in den einzelnen Partitionen ist ein vielversprechender Kandidat für eine Optimierung der Ausführungszeit.

Auch der Klassifikations-Workflow, der in Abbildung 7.1 dargestellt wird, ist zunächst eine sequentielle Abfolge von abstrakten Arbeitsschritten. Eine parallele Berechnung voneinander unabhängiger Arbeitsschritte ist hier eine naheliegende Optimierung. Nach dem *Gesetz von Amdahl* [Amdahl (1967)] hängt der *SpeedUp*, den ein Programm durch Parallelisierung erfahren kann, jedoch direkt von dem prozentualen Anteil paralleler Programmabschnitte ab. Allerdings gestaltet sich eine Parallelisierung der einzelnen Arbeitsschritte schwierig, da der Nutzer diese erst zur Laufzeit in Abhängigkeit von der zu untersuchenden Domäne konkretisiert. Um nun einzelne abstrakte Arbeitsschritte unabhängig voneinander ausführen zu können, ohne ihre konkrete Umsetzung zu kennen, wird auf die monadische (nach Definition 18 auf Seite 69) `Future`-Implementierung des Akka-Frameworks zurückgegriffen, wie sie in Abschnitt 6.5 beschrieben wird.

Die vorherigen Abschnitte haben die verschiedentlichen Arbeitsschritte und resultierenden Zwischenformate beschrieben, die das Workflow für die Erstellung des indexierten Datenformats vorschreibt, anhand dessen anschließend die Klassifikationsläufe vorgenommen werden können. Um diesen Vorgang zu beschleunigen, können voneinander unabhängige Arbeitsschritte parallel ausgeführt werden. Dabei muss berücksichtigt werden, dass alle asynchronen, abstrakten Arbeitsschritte abgeschlossen sein müssen, bevor das Endresultat erzeugt werden kann.

Code-Beispiel 7.5 verdeutlicht dieses Prinzip: Zunächst werden zwei Indices angelegt, Satz- und Wort-Index, wobei die Dauer und Implementierung dieser Vorgänge jedoch unbestimmt ist. Anstatt nun aber blockierend zu warten, bis die Erzeugung beider Indices abgeschlossen ist, wird definiert, dass bei Abschluss beider asynchroner Aufgaben die nächste, ebenfalls asynchrone Aktion angestoßen werden soll. In diesem Fall ist das die Erzeugung des invertierten Indexes, der als Vorbedingung sowohl einen Satz- als auch einen Wort-Index benötigt.

Die monadischen Futures erweitern somit die Basisttypen, welche die Zwischenergebnisse des abstrakten Workflows darstellen, einerseits um die Fähigkeit, erst zu einer unbestimmten Zeit in der Zukunft mit einem Wert belegt zu sein, andererseits ermöglichen sie eine Komposition mehrerer unabhängiger Futures zu einer abstrakten Verarbeitungskette. Im nachfolgenden Abschnitt werden die daraus re-

sultierenden Verbesserungen der Laufzeit exemplarisch belegt.

```
0  // -- create indexes
1
2  // sentence index
3  final Future<Boolean> fSentence = createSentenceIndex(s, source, sOut,
4      ec);
5
6  // word index
7  final Future<Boolean> fWord = createWordIndex(p, source, wOut, ec);
8
9  // (wait for completion)
10 final List<Future<Boolean>> indexFutures = new ArrayList<>();
11 indexFutures.add(fSentence);
12 indexFutures.add(fWord);
13 final Future<Iterable<Boolean>> indexFuturesCompleted = sequence(
14     indexFutures, ec);
15
16 // inverted index
17 final Future<Boolean> result = createInvertedIndex(
18     indexFuturesCompleted, sOut, wOut, i, iOut, p, ec);
19
20 // create vector index and shutdown
21 final Future<Boolean> finalResult = createVectorDB(result, label, c,
22     cOut, ec).andThen(shutdown(ec), ec);
23
24 return Await.result(finalResult, TIMEOUT);
```

Listing 7.5: Komposition von Akka Futures

Evaluation

In diesem Abschnitt sollen einige beispielhafte Konfigurationen und die anschließende Verwendung des Klassifikations-Frameworks gezeigt werden. Dabei sollen insbesondere die Effektivität der Parallelisierung unabhängiger Arbeitsschritte sowie die leichte Konfiguration der Klassifikations-Parameter belegt werden. Das Kapitel schließt mit beispielhaften Ausgaben und deren Verwertungsmöglichkeiten.

Anwendungsbeispiele und Ausgabe

In Code-Beispiel 7.4 (Seite 83) wurde zuvor bereits eine beispielhafte Konfigurations-Klasse für das Klassifikations-Framework gezeigt. Diese Konfiguration kann nun leicht mittels Vererbung modifiziert werden, indem man sie in einer eigenen Unterklasse erweitert.

```
0  public class ParallelQueryConfig extends BasicConfig {
1    public ParallelQueryConfig() throws IOException {
2      super();
3    }
4
5    @Override
6    @Bean
7    public Query query() {
8      return new ActorQuery();
9    }
10 }
```

Listing 7.6: Anpassen der Konfiguration durch Vererbung

Code-Beispiel 7.6 zeigt eine Klasse, welche von der Basis-Konfiguration erbt und eine alternative, parallelisierte Abfrage-Komponente spezifiziert. Anschließend muss diese modifizierte Konfiguration geladen und die am Workflow beteiligten Objekte erzeugt werden. Code-Beispiel 7.7 illustriert diesen Vorgang.

```
0      // create project setup
1      final AnnotationConfigApplicationContext ctx
2        = new AnnotationConfigApplicationContext();
3      ctx.register(ParallelQueryConfig.class);
4      ctx.refresh();
5
6      // get classification workflow components
7      final IndexRepository repo = ctx.getBean(IndexRepository.class);
8      final IndexSetup idxSetup = ctx.getBean(IndexSetup.class);
9      final QuerySetup querySetup = ctx.getBean(QuerySetup.class);
```

Listing 7.7: Laden einer Konfigurations-Klasse

Wie in Bild 7.1 auf Seite 73 bereits gezeigt, ist der Klassifikations-Workflow zweigeteilt: Zunächst müssen die notwendigen Datenstrukturen erzeugt werden, anschließend kann das Framework Klassifikationsläufe vornehmen.

In Code-Beispiel 7.8 wird gezeigt, wie die Erstellung der Datenstrukturen für den Klassifikationsvorgang initiiert wird, wobei die zuvor mit Hilfe der Konfigurations-Klasse erzeugten Module verwendet werden. Zu sehen ist die monadische Verkettung von Arbeitsschritten insofern, als dass nach Abschluss des sowohl bezüglich der Dauer als auch der konkreten Implementierung undefinierten Vorgangs ein weiterer asynchroner Vorgang angestoßen wird, nämlich das Postprocessing und Terminieren des Programms in der `shutdown()`-Methode. Besonders zu betonen

ist hierbei, dass der in diesem Beispiel angesprochene Vorgang der Erstellung der indexierten Datenstrukturen in sich ebenfalls in mehrere abstrakte und asynchron abgearbeitete Einzelschritte aufgeteilt ist, wie es in Abschnitt 7.3 beschrieben und in Code-Beispiel 7.5 veranschaulicht wurde.

```
0   private static void createIndex(final IndexRepository repo,
1       final IndexSetup setup, final ExecutionContextExecutorService ec)
2   {
3     // create new index
4     repo.createIndex(setup, ec)
5     // shutdown after completion
6         .andThen(shutdown(ec), ec);
7   }
```

Listing 7.8: Erstellen der Index-Strukturen

Nachdem die notwendigen Datenstrukturen erzeugt wurden, kann anschließend eine Klassifikations-Anfrage gestellt werden. Code-Beispiel 7.9 zeigt, wie dafür zunächst der Referenz-Vektor aufgebaut (anhand der Terme *„london“*, *„schlieren“*, *„bayern“*, *„usa“*) und dieser dann mit der Vektor-Datenbank verglichen wird.

```
0   private static void query(final QuerySetup setup, final String label)
1       throws Exception
2   {
3     // query
4     final Query q = setup.queryByLabel(label);
5     final Set<String> test = new LinkedHashSet<>();
6     Collections.addAll(test,
7         new String[] { "london", "schlieren", "bayern", "usa"});
8     final Map<String, Double> result = q.findSimilar(0.90d, test);
9
10    // handle results ...
11  }
```

Listing 7.9: Erstellung eines Referenz-Vektors und Klassifikationslauf

Das Ergebnis umfasst diejenigen Vektoren, die den gegebenen Ähnlichkeits-Schwellwert (im Beispiel 0.9) überschreiten. Code-Beispiel 7.10 zeigt eine Auflistung der Terme und ihrer jeweiligen berechneten Ähnlichkeitswerte zum Referenzvektor.

```
0  Identified 'betracht' with similarity of '0.9504774623040696'.
1  Identified 'voltlage' with similarity of '0.9051066306401438'.
2  Identified 'hettiswil' with similarity of '0.9115726891665564'.
3  Identified 'basel' with similarity of '0.9314396042381823'.
4  Identified 'arnsdorf' with similarity of '0.9092288780112022'.
5  Identified 'vorauseilen' with similarity of '0.9049000067302381'.
6  Identified 'frankfurt' with similarity of '0.9733773401544089'.
7  Identified 'bangkok' with similarity of '0.9723716944011446'.
8  Identified 'bielefeld' with similarity of '0.937394098200564'.
9  Identified 'vancouver' with similarity of '0.9235873006539592'.
10 Identified 'ostheim' with similarity of '0.924689122275267'.
11 Identified 'einzelf llen' with similarity of '0.9161434467511563'.
12 Identified 'schach' with similarity of '0.9278663631140475'.
13 Identified 'neon' with similarity of '0.9068712927837272'.
```

Listing 7.10: Ausgabe des Klassifikations-Frameworks zu Beispiel 1 (Auszug)

Belege für Effizienz

Abbildung 7.5 zeigt eine Gegenüberstellung des sequentiellen und parallelen Workflows. Durch die monadische Komposition und parallele Ausführung unabhängiger Arbeitsschritte konnte die Laufzeit des Klassifikations-Workflows halbiert werden. Dieser Effekt wird bei sukzessiven Abfragen auf der gleichen Vektor-Datenbank noch deutlicher, da die Zeit für das Laden der indexierten Datenstruktur dann entfällt.

Dieses Kapitel hat die Implementierung und die Verwendung des Klassifikations-Frameworks demonstriert. Dabei wurde besonders die Verknüpfung asynchroner, abstrakter Arbeitsschritte illustriert, durch deren unabhängige Berechnung die Laufzeit des Workflows bereits auf Framework-Ebene reduziert werden konnte, ohne auf die vom Nutzer gewählten konkreten Arbeitsschritte eingehen zu müssen. Gezeigt wurde auch die leichte Anpassbarkeit des Klassifikations-Verhaltens über Konfigurations-Klassen, was eine hohe Flexibilität bei sich ändernden Domänen gewährleistet. Das nachfolgende Kapitel soll nun die Leistung der bisher beschriebenen Ansätze evaluieren.

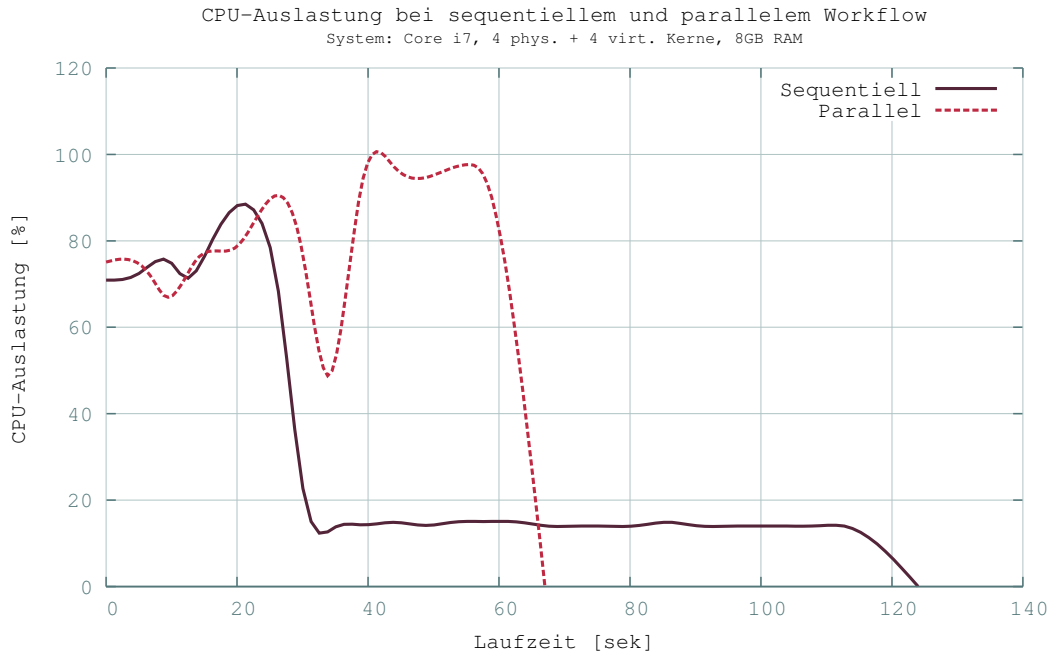


Abbildung 7.5: Laufzeit und CPU-Auslastung bei sequentiellem und parallelem Workflow.

8 Evaluation

Das folgende Kapitel evaluiert die Leistungsfähigkeit in dieser Arbeit vorgestellten Ansätze zur Metadaten-Extraktion in historischen Texten. Zunächst werden dafür die Qualitätsmaße definiert und die Test-Szenarien beschrieben. Danach folgt eine Gegenüberstellung der Leistungen der jeweiligen Ansätze und eine Interpretation der Resultate.

Qualitätsmaße

Für Klassifikationsaufaben wird die Qualität des Klassifikators üblicherweise anhand einer *Konfusionsmatrix* abgeleitet. Diese Matrix kann erstellt werden, wenn Wissen über die korrekte Klassifikation hinsichtlich einer Eigenschaft bekannt ist und die Vorhersagen des Klassifikators daran gemessen werden können.

	$\alpha = 1$	$\alpha = 0$
$a = 1$	TP	FP
$a = 0$	FN	TN

Tabelle 8.1: Konfusionsmatrix

Sei α die Vorhersage des Klassifikators bezüglich der Klassenzugehörigkeit eines Objekts und a die tatsächliche Klassenzugehörigkeit, dann ergeben sich die in Tabelle 8.1 gezeigten Konstellationen: Man unterscheidet zwischen *true positives (TP)*, den vom Klassifikator richtig erfolgten Zuordnungen zu einer Klasse, und *true negatives (TN)*, den vom Klassifikator richtig als nicht zu einer Klasse gehörend erkannten Instanzen. Weiterhin gibt es die beiden Fehlerfälle der *false*

positives (FP), bei denen der Klassifikator fälschlicherweise eine Zuordnung zu einer Klasse vorhergesagt hat, und analog den *false negatives (FN)*, die diejenigen Instanzen enthalten, die vom Klassifikator nicht als zu einer Klasse zugehörig erkannt wurden, es allerdings sind.

Aus den vier Werten einer Konfusionsmatrix können dann Verhältnisse abgeleitet werden, die sich zur Leistungsbewertung eines Klassifikators eignen (so definiert beispielsweise in [Ferber (2003)]).

Definition 19. Unter dem Begriff **precision** versteht man die Genauigkeit eines Klassifikators. Er beschreibt das Verhältnis zwischen den tatsächlich zu einer Klasse gehörenden Instanzen *(TP)* zu den vom Klassifikator vermuteten Instanzen dieser Klasse, welches sowohl richtige *(TP)* als auch falsche *(FP)* Vorhersagen enthalten kann.

$$precision = \frac{TP}{TP + FP} \tag{8.1}$$

Definition 20. Der Begriff **recall** steht für die Empfindlichkeit oder Trefferquote eines Klassifikators und gibt den Anteil der vom Klassifikator korrekt erkannten Instanzen *(TP)* zu den tatsächlich korrekten Instanzen an *(TP + FN)*.

$$recall = \frac{TP}{TP + FN} \tag{8.2}$$

Definition 21. Im Gegensatz zu precision und recall, welche nur im Verbund zur Bewertung eines Klassifikators geeignet sind, erlaubt es das F_1**-Maß**, die Leistung eines Klassifikators mit einer einzigen Kennzahl zu beziffern. Dazu kombiniert es precision und recall über Bildung des gewichteten harmonischen Mittels.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{8.3}$$

Corpora

In diesem Abschnitt werden die Corpora vorgestellt, anhand derer die Leistungsbewertung der Ansätze vorgenommen werden soll.

Definition 22 (CorpusMerian)**.** Für die Tests auf historischen Texten wurde das Werk *Topographia Franconia* (nachfolgend als **CorpusMerian** bezeichnet) von *Matthäus Merian (dem Älteren)* [Merian (1642)] aus dem Jahr 1648 gewählt. Dieser Band aus dem Hauptwerk *Topographia Germaniae* umfasst in etwa 76.000 Worte in Frühneuhochdeutsch und beschreibt detailliert Städte und Gemeinden aus dieser Zeit. Aufgrund der enzyklopädischen Natur dieses Werkes eignet es sich besonders für die Leistungsbewertung von Klassifikatoren für Toponyme. Eine digitalisierte Version kann unter *Franconica Online*[1] eingesehen werden.

Definition 23 (CorpusLeipzig)**.** Zusätzlich zum historischen Text wurde ein Corpus der Universität Leipzig (wie beschrieben in [Quasthoff, Richter und Biemann (2006)], nachfolgend als **CorpusLeipzig** bezeichnet) verwendet. Die in mehreren Sprachen erhältlichen Corpora der Universität Leipzig enthalten einzelne, zufällig gewählte Sätze, wobei als Quellen Zeitschriften, Zeitungen und im Internet publizierte Sätze genannt werden[2]. Die Corpora sind in unterschiedlichen Größen erhältlich und enthalten zwischen zehntausend und einer Million Satz-Instanzen.

Für die nachfolgenden Tests wurde ein deutschsprachiger Corpus mit 100.000 Sätzen gewählt. Er stellt durch den um viele Jahrhunderte neueren Sprachgebrauch, eine andere inhaltliche Ausrichtung, Spezifität und Domäne der Quelltexte eine kontrastreiche Alternative zu *CorpusMerian* dar, sodass er zur Validierung der Testresultate gut geeignet ist.

[1] http://franconica.uni-wuerzburg.de/ub/topographia-franconiae/merian/meriancover.html
[2] http://corpora.informatik.uni-leipzig.de/download.html

Testmethodik

Die Leistungstests sollen zeigen, wie gut sich die vorgestellten Ansätze zur Klassifikation von Toponymen eignen. Für die Auswertung der Klassifikationsleistung wurden die Corpora dazu in Trainings- und Testmengen unterteilt. Als Testmenge wurden jeweils 1.000 zufällig aus dem ursprünglichen Corpus entfernte Sätze gewählt und der verbleibende Text als Trainingsmenge verwendet. In der Testmenge wurden dann alle Toponym-Funde annotiert, wobei nicht nur klassischerweise Ortsnamen als Treffer markiert wurden, sondern darüber hinaus auch Toponyme von höherer Granularität (Länder, Kontinente) bis hin zu Toponymen von abstrakter oder konzeptioneller Natur (Beispiel „am Brunnen“). Die Referenz-Terme, die der Klassifikations-Algorithmus als Eingabe erwartet, wurden ihrerseits zufällig aus den annotierten Treffern gewählt. Die Leistungsmaße precision, recall und F_1-Maß werden jeweils in Abhängigkeit zum Schwellwert der Distanzmetrik angegeben. Die in den folgenden Leistungstests verwendete Distanzmetrik ist das Cosinus-Maß (wie definiert in Abschnitt 5.2.5). Weitere Tests und Resultate finden sich in [Schneider (2013)].

8.1 Leistungsbewertung

Die nachfolgenden Abschnitte beschreiben die Resultate von Testläufen der vorgestellten Ansätze auf unterschiedlichen Corpora bei wechselnden Konfigurationen.

Testszenario 1: Kontext-Vektor Retrieval, neusprachlicher Corpus

Als Ausgangspunkt soll zunächst die Leistung des Klassifikations-Frameworks, wie es in Abschnitt 5.2 auf Seite 41 definiert wurde, bestimmt werden. Folgende Konfiguration kam dabei zum Einsatz:

Test-Parameter	Wert
Quelldokument	CorpusLeipzig
gesuchte Klasse	Toponyme
Verfahren	Kontext-Vektor Retrieval
Feature-Module	Kontext-Komposition, Co-Occurrence Daten

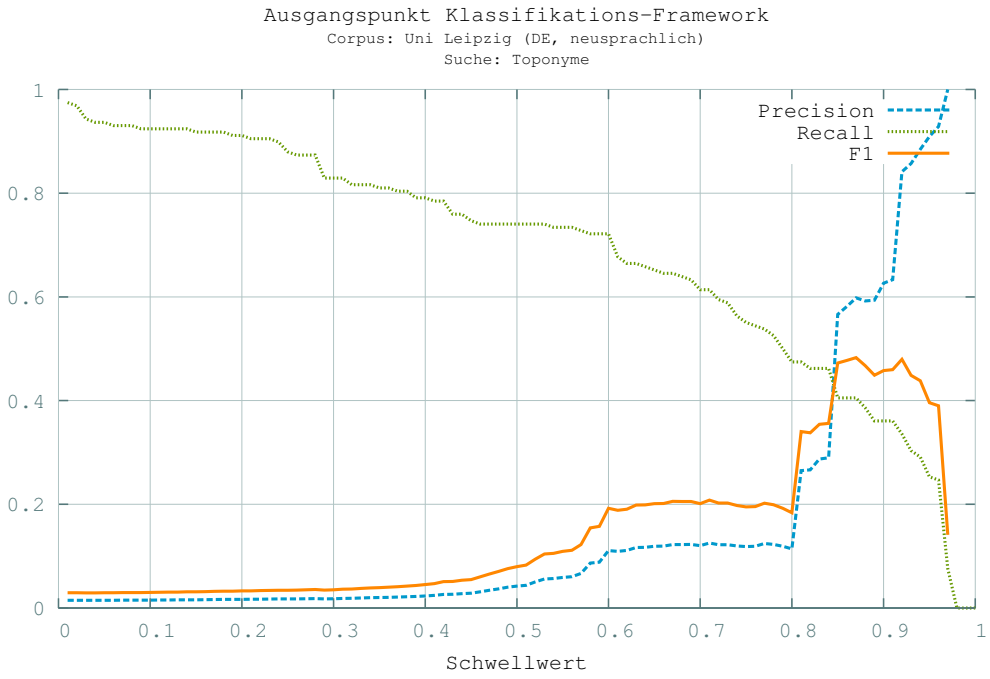


Abbildung 8.1: Testszenario 1, Kontext-Vektor Retrieval, Quelldokument: CorpusLeipzig.

Auswertung Abbildung 8.1 zeigt das Resultat des Testszenarios des Klassifikations-Frameworks unter Verwendung der Kontext-Vektoren, wie sie in Kapitel 5.2.6 beschrieben werden. Dabei sind nur grundlegende statistische Feature-Module (`ContextComposition` und `ContextCompositionDistance`, wie beschrieben in Abschnitt 5.2.3) zur Klassifikation herangezogen worden. Insbesondere fällt auf, dass die Genauigkeit des Klassifikators mit zunehmendem Ähnlichkeits-Schwellwert δ lange stagniert und erst bei Werten $\delta \geq 0.8$ sprunghaft ansteigt.

Trotz der orthographischen Konsistenz des neusprachlichen Corpus überschreitet das F_1-Maß einen Wert von etwa 0.45 nicht.

Testszenario 2: Kontext-Vektor Retrieval, historischer Corpus

Die Leistung des Klassifikators soll nun auch auf historischen Texten beurteilt werden. Folgende Parameter wurden für das Testszenario verwendet:

Test-Parameter	Wert
Quelldokument	CorpusMerian
gesuchte Klasse	Toponyme
Verfahren	Kontext-Vektor Retrieval
Feature-Module	Kontext-Komposition, Co-Occurrence Daten

Auswertung Abbildung 8.2 zeigt das Resultat von Testszenario 2. Ähnlich wie beim neusprachlichen Corpus im vorherigen Test überschreitet das F_1-Maß einen Wert von ungefähr 0.45 nicht. Die Leistung des Klassifikators ist zwar trotz des Domänenwechsels stabil, kann jedoch noch nicht überzeugen. Auffällig ist, dass der maximale Wert des F_1-Maßes bei einem deutlich niedrigeren Ähnlichkeits-Schwellwert erreicht wird, als es bei dem neusprachlichen Text der Fall war. Nachfolgend sollen nun unterschiedliche Parameter der Testszenarien sowie die in Abschnitt 5.3 angesprochenen Maßnahmen zur Optimierung der Klassifikations-Leistung evaluiert werden.

Testszenario 3: Kontext-Vektor Retrieval, neusprachlicher Corpus, modifizierte Referenzmenge

In Abschnitt 5.3.1 wird als Maßnahme zur Optimierung der Klassifikations-Qualität besprochen, die Menge der Referenz-Terme für das Kontext-Vektor Retrieval zu erhöhen. Daher soll in diesem Testszenario evaluiert werden, welchen Einfluss die Anzahl der Referenz-Terme auf die Leistung des Klassifikators besitzt.

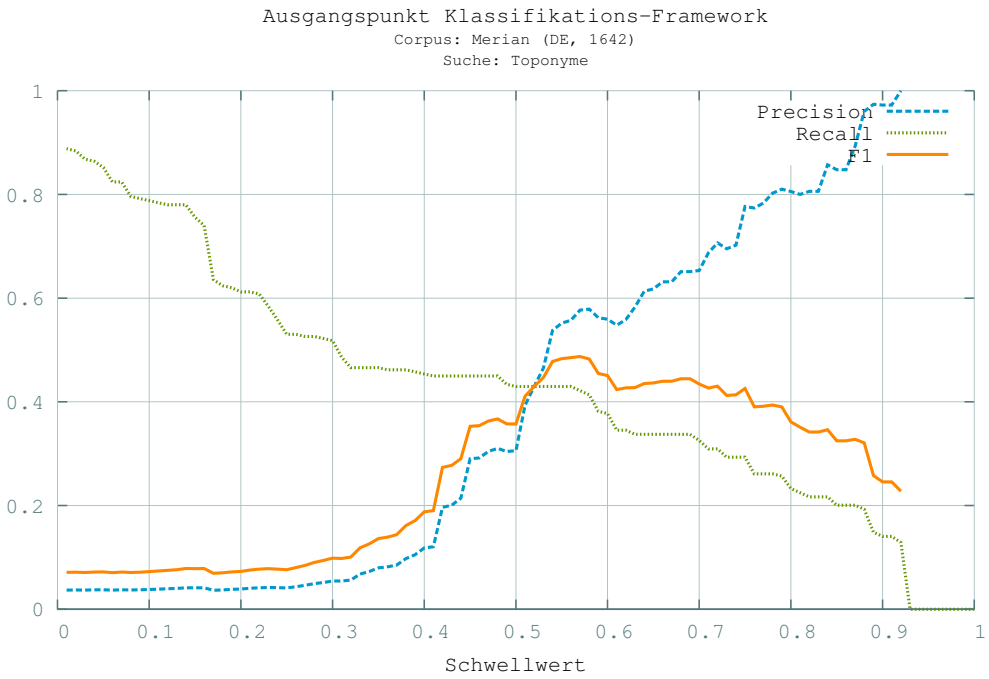


Abbildung 8.2: Testszenario 2, Kontext-Vektor Retrieval, Quelldokument: CorpusMerian.

Test-Parameter	Wert
Quelldokument	CorpusLeipzig
gesuchte Klasse	Toponyme
Verfahren	Kontext-Vektor Retrieval
Feature-Module	Kontext-Komposition, Co-Occurrence Daten
Optimierung	sukzessive Erweiterung der Referenz-Term-Menge

Auswertung Abbildung 8.3 zeigt, dass bereits nach ungefähr 30 Beispielen die Leistungsparameter des Klassifikators konvergieren und das Hinzufügen weiterer Instanzen keine Auswirkungen hat.

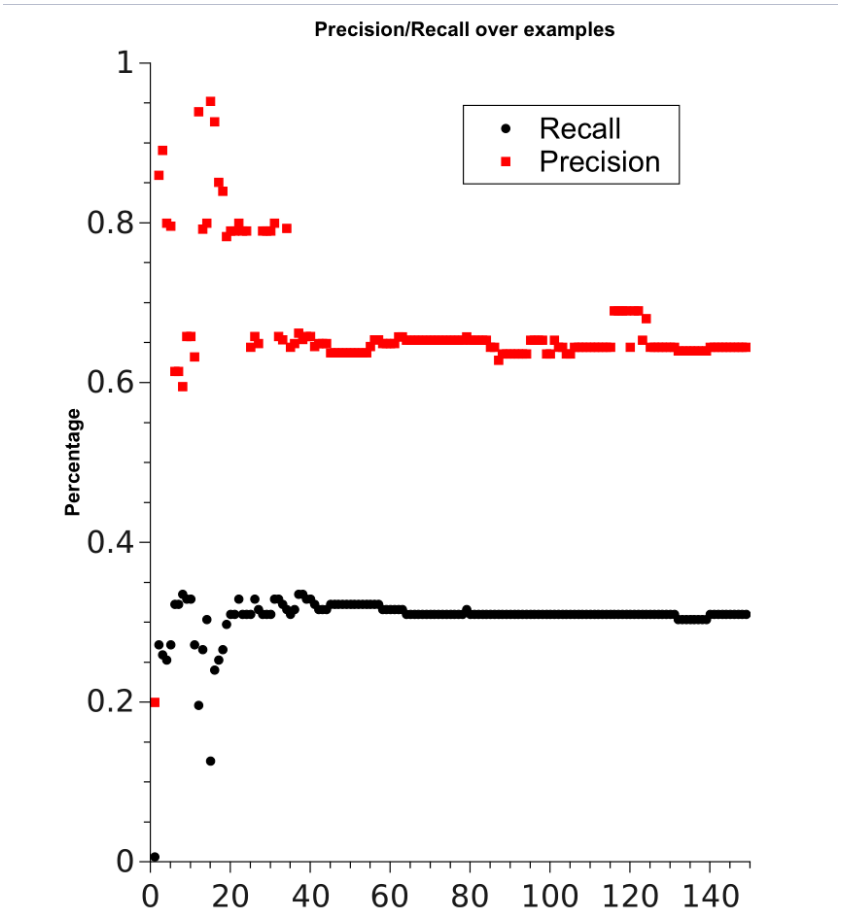


Abbildung 8.3: Testszenario 3, Perzeptron, Klassifikationsleistung gegen Anzahl Referenzterme, Quelldokument: CorpusLeipzig, Quelle: [Schneider (2013)].

Testszenario 4: Perzeptron, neusprachlicher Corpus

Testszenario 4 soll die Klassifikationsleistung des auf dem Perzeptron basierenden Ansatzes evaluieren, bevor der Effekt weiterführender Optimierungen beurteilt werden kann. Testszenario 4 sieht folgende Parameter vor:

Test-Parameter	Wert
Quelldokument	CorpusLeipzig
gesuchte Klasse	Toponyme
Verfahren	Perzeptron
Feature-Module	Kontext-Komposition, Co-Occurrence Daten

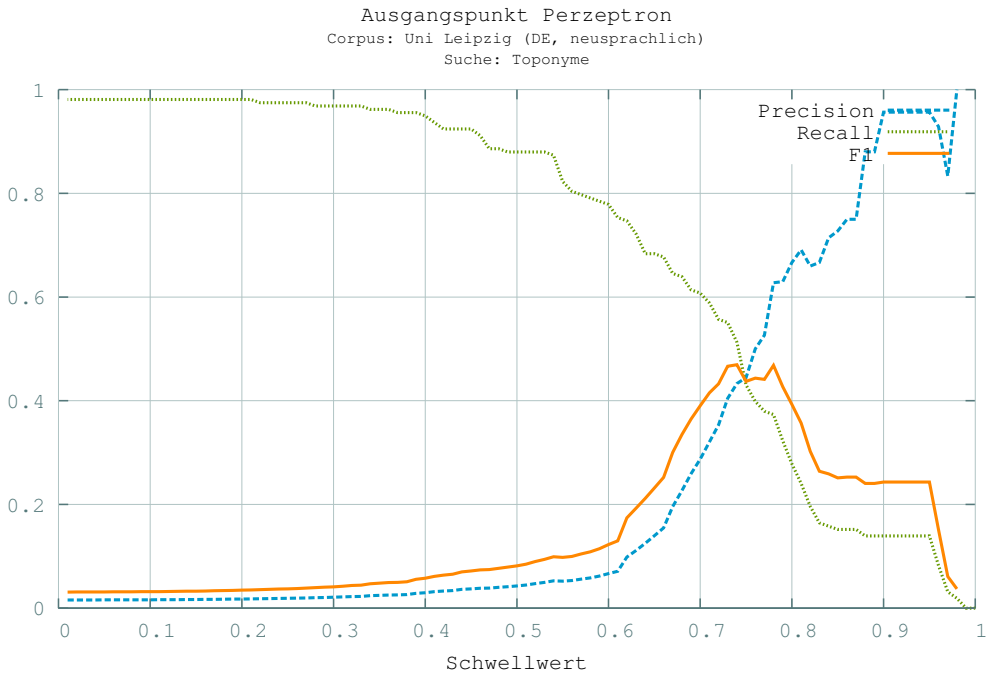


Abbildung 8.4: Testszenario 4, Perzeptron, Quelldokument: CorpusLeipzig.

Auswertung Die Leistung des Verfahrens entspricht mit einem F_1-Maß von etwa 0.45 der Leistung des Kontext-Vektor Retrievals. Jedoch wird der maximale F_1-Wert bereits bei einem Schwellwert von ungefähr $\delta = 0.7$ erreicht.

Testszenario 5: Perzeptron, historischer Corpus

In diesem Szenario soll die Ausgangsleistung des Perzeptron-Klassifikators auf dem historischen Material evaluiert werden:

Test-Parameter	Wert
Quelldokument	CorpusMerian
gesuchte Klasse	Toponyme
Verfahren	Perzeptron
Feature-Module	Kontext-Komposition, Co-Occurrence Daten

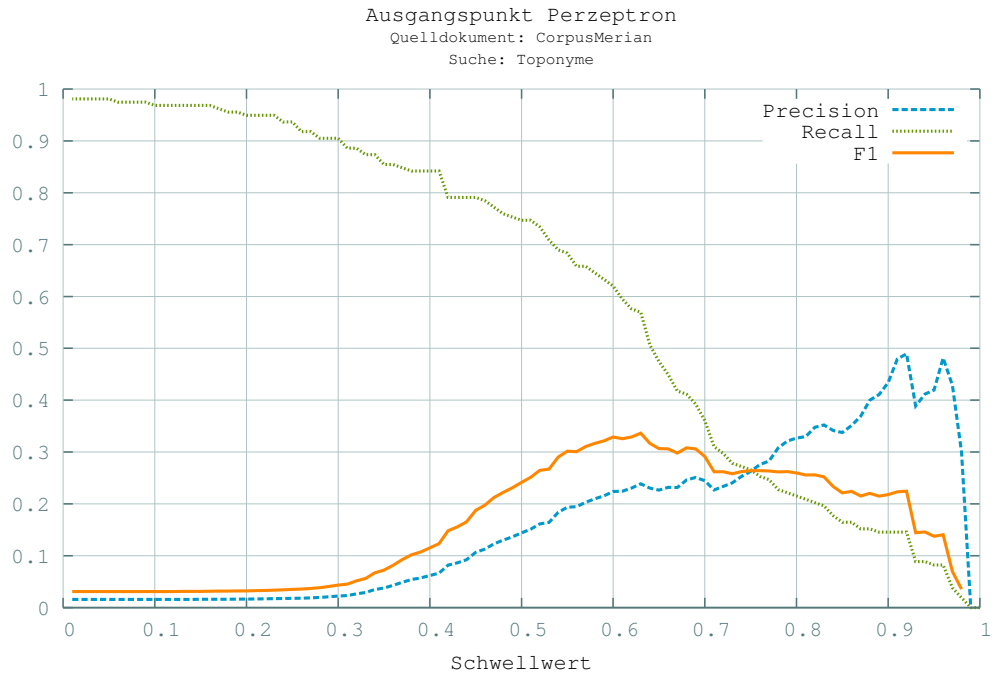


Abbildung 8.5: Testszenario 5, Perzeptron, Quelldokument: CorpusMerian.

Auswertung Abbildung 8.5 belegt die schlechtere Klassifikations-Leistung auf dem historischen Dokument: Der maximale Wert des F_1-Maßes liegt bei etwa 0.32, wird jedoch bereits ab einem Schwellwert von $\delta = 0.6$ erreicht.

Testszenario 6: Perzeptron, neusprachlicher Corpus, Erweiterung der Referenz-Menge

Wie zuvor in Testszenario 3 soll nachfolgend überprüft werden, ob die Hinzunahme weiterer Referenz-Terme Vorteile für die Klassifikation durch das Perzeptron mit sich bringt.

Test-Parameter	Wert
Quelldokument	CorpusLeipzig
gesuchte Klasse	Toponyme
Verfahren	Perzeptron
Feature-Module	Kontext-Komposition, Co-Occurrence Daten
Optimierung	sukzessive Erweiterung der Menge der Referenz-Terme

Auswertung Im Gegensatz zu dem in Testszenario 3 untersuchten Verfahren, welches mit Kontext-Vektor Retrieval arbeitet, profitiert das auf dem Perzeptron basierende Verfahren von den zusätzlichen Referenz-Termen. Abbildung 8.6 zeigt die deutliche Verbesserung der Kenngrößen.

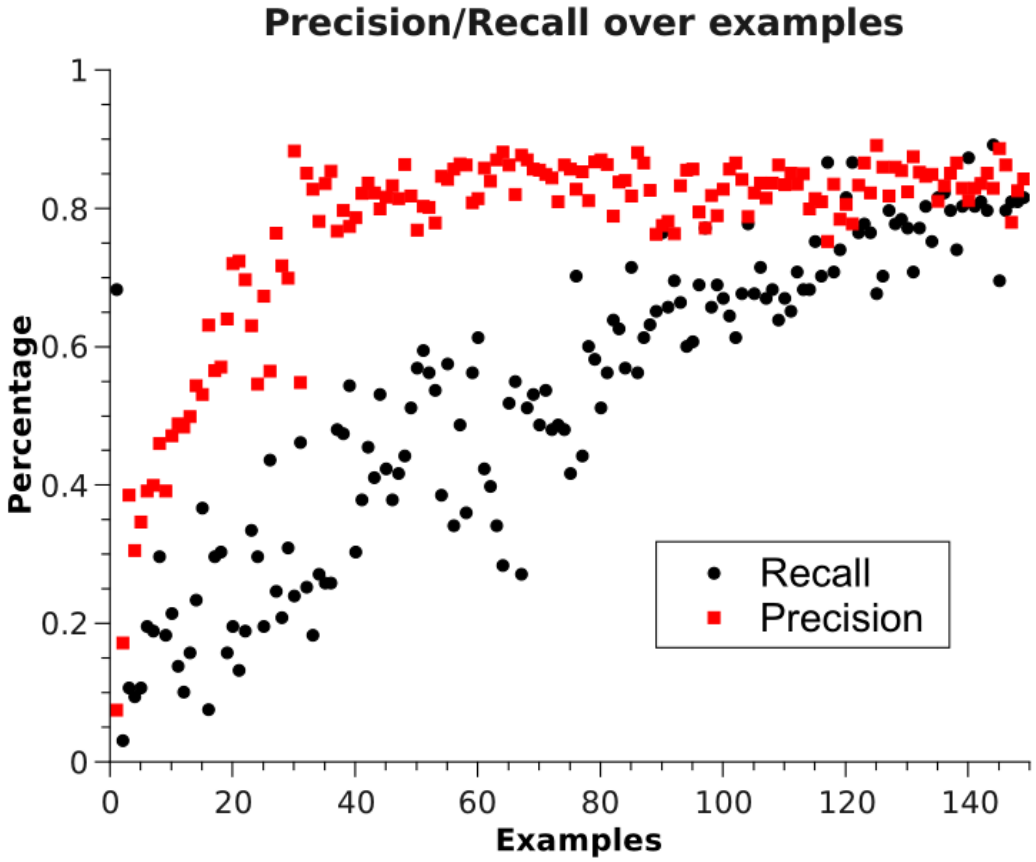


Abbildung 8.6: Testszenario 6, Perzeptron, Precision und Recall gegen Anzahl Referenzterme, Quelldokument: CorpusLeipzig, Quelle: [Schneider (2013)].

Testszenario 7: Optimiertes Perzeptron

In Kapitel 5 wurde auf den negativen Effekt verwiesen, den die Klassifikations-Leistung durch zu viele, für den Klassifikations-Prozess irrelevante Dimensionen erfährt. Als Gegenmaßname soll ein Gewichtungs-Modell verwendet werden, welches häufig vergebene Features des Kontext-Vektors mit einem hohen Gewicht und seltene mit einem niedrigen versieht. In diesem Testszenario sollen nur diejenigen Dimensionen des Kontext-Vektors zur Klassifikation herangezogen werden, deren Gewicht über einem gewissen Schwellwert liegt.
Weiterhin wurde die Topologie des Perzeptrons modifiziert, sodass es nun mehrere Klassen klassifizieren kann. Weitere Informationen zu der Modifikation finden sich in [Schneider (2013)].

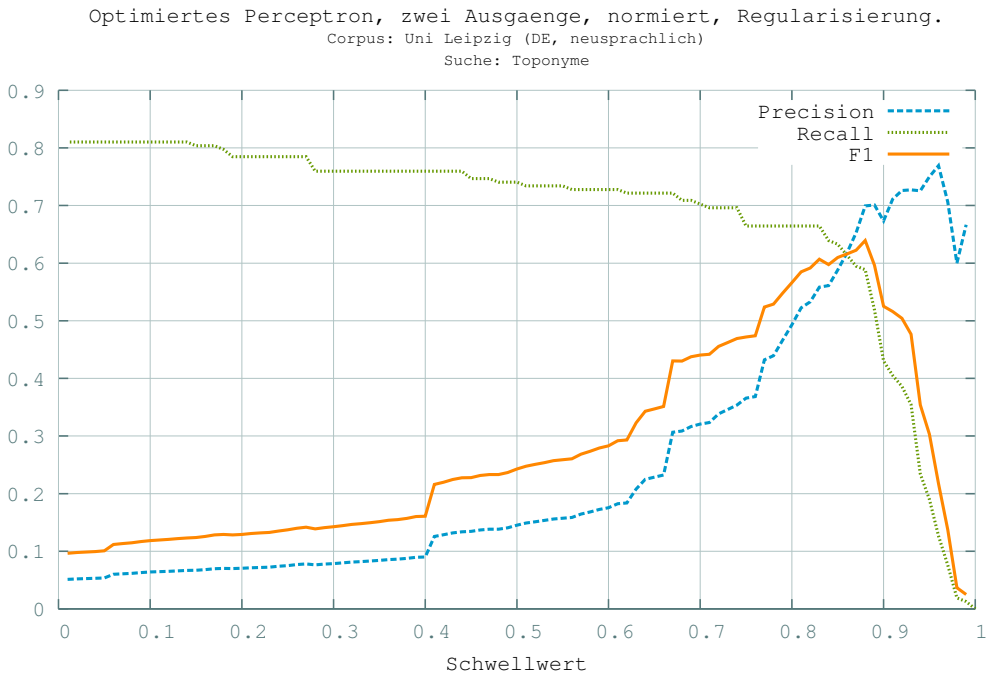


Abbildung 8.7: Testszenario 7, Optimiertes Perzeptron, Quelldokument: CorpusLeipzig.

Auswertung Abbildung 8.7 zeigt eine deutliche Verbesserung der Leistungsgrößen. Insbesondere die Reduktion der irrelevanten Features im Kontext-Vektor hat sich als effektive Maßnahme zur Steigerung der Klassifikations-Leistung herausgestellt.

Testszenario 8: Kontext-Vektor Retrieval mit optimiertem Perzeptron, Boosting, neusprachlicher Corpus

Dieses Testszenario kombiniert das optimierte Perzeptron des vorherigen Testszenarios mit dem Kontext-Vektor Retrieval zu einem Verbundsklassifikator. Jeder beteiligte Klassifikator geht dabei mit einem Gewicht in die Verbundsaussage ein. Das entspricht dem Boosting-Verfahren, wie es in Abschnitt 5.3.1 vorgestellt wurde.

Test-Parameter	Wert
Quelldokument	CorpusLeipzig
gesuchte Klasse	Toponyme
Verfahren	Kontext-Vektor Retrieval und optimiertes Perzeptron
Feature-Module	Kontext-Komposition, Co-Occurrence Daten, Co-Occurrence Daten mit Distanzfunktion

Auswertung Abbildung 8.8 zeigt einerseits ein deutlich besseres Ergebnis für die Klassifikationsleistung des Verbundsklassifikators im Vergleich zu den Testszenarien 1 und 4, in denen die beteiligten Verfahren einzeln getestet wurden. Das F_1-Maß erreicht einen maximalen Wert von 0.66. Dabei fällt auch der homogenere Verlauf des F_1-Maßes über ein breiteres Intervall des Ähnlichkeits-Schwellwertes hinweg auf. Mit Hilfe des Boosting-Verfahrens können die Vorzüge beider vorgestellter Verfahren ausgenutzt und die Klassifikations-Qualität gesteigert werden.

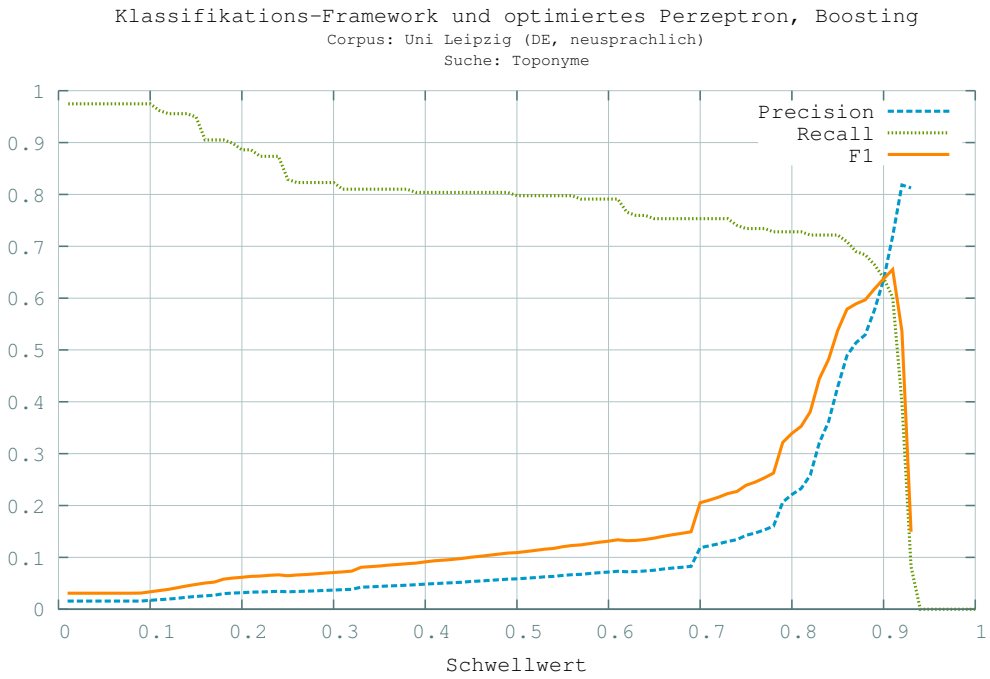


Abbildung 8.8: Klassifikations-Framework und optimiertes Perzeptron, Boosting, Quelldokument: CorpusLeipzig.

Testszenario 9: Kontext-Vektor Retrieval mit optimiertem Perzeptron, Boosting, historischer Corpus

Auch auf historischen Texten soll der Verbundsklassifikator, der aus dem optimiertem Perzeptron und dem Kontext-Vektor Retrieval durch nachgelagertes Boosting gewonnen wird, getestet werden. Testszenario 9 hat folgende Parameter:

Test-Parameter	Wert
Quelldokument	CorpusMerian
gesuchte Klasse	Toponyme
Verfahren	Kontext-Vektor Retrieval und optimiertes Perzeptron
Feature-Module	Kontext-Komposition, Co-Occurrence Daten, Co-Occurrence Daten mit Distanzfunktion

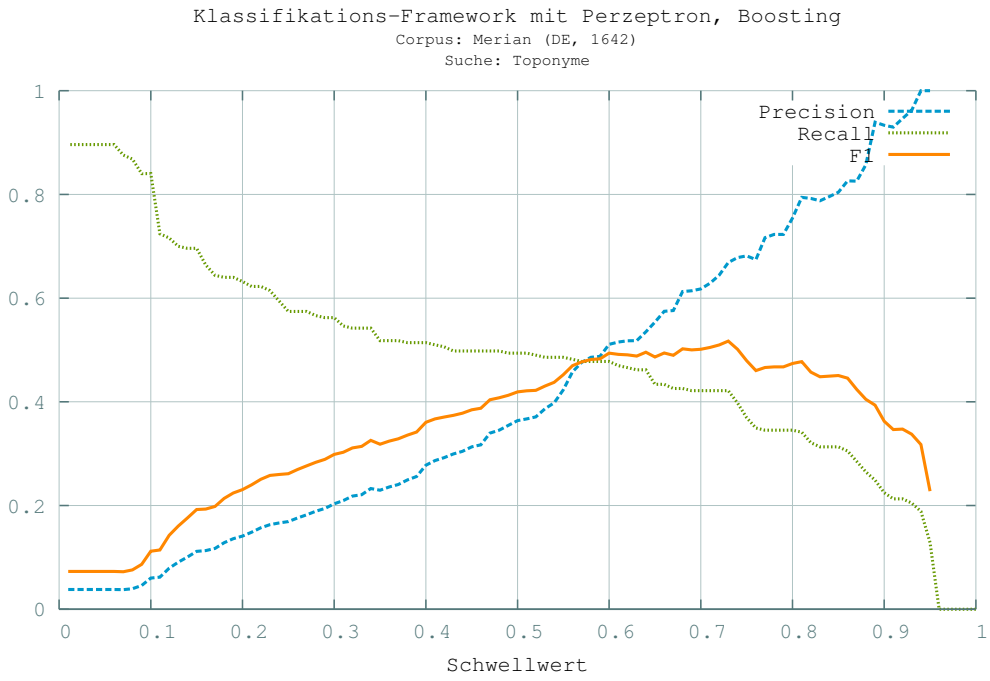


Abbildung 8.9: Klassifikations-Framework und optimiertes Perzeptron, Boosting, Quelldokument: CorpusMerian.

Auswertung Wie auch das vorherige Testszenario kann Testszenario 9 den Wert des F_1-Maßes auf 0.55 anheben. Ebenfalls ist wieder der deutlich homogenere Werte-Verlauf für F_1 über ein breiteres Schwellwert-Intervall zu sehen.

Fazit

Die in diesem Kapitel vorgestellten Testszenarien belegen, dass sich die in dieser Arbeit besprochenen Ansätze dazu eignen, Klassen wie Toponyme in Texten zu finden. Dabei bleibt die Klassifikations-Qualität auch dann stabil, wenn die Domäne der untersuchten Texte sich ändert oder sich die Texte durch eine herausfordernde Syntax auszeichnen.
Insbesondere verdeutlichen die Resultate auch die Wirksamkeit der Maßnahmen

zur Optimierung der Klassifikations-Qualität, wie sie in Kapitel 5.3 besprochen wurden. Als besonders effektiv stellt sich dabei heraus, die Dimensionszahl des Kontext-Vektors zu reduzieren, indem die für den Klassifikations-Vorgang irrelevanten Dimensionen entfernt werden. Der Effekt des „Fluchs der Dimensionalität“ wird somit reduziert.
Die Verwendung des nachgeschalteten Boostings erlaubt es nicht nur, die Vorzüge aller beteiligten Klassifikations-Verfahren ausnutzen und die Klassifikations-Leistung über ein harmonischer verlaufendes, breiteres Schwellwert-Intervall abrufen zu können, sie erleichtert auch die zukünftige Erweiterung des Verfahrens um weitere Klassifikatoren.
Da es sich bei den vorgestellten Ansätzen um Verfahren aus dem *Unsupervised Learning* handelt, eignen sie sich insbesondere für den Einsatz in einer automatisierten Verarbeitungskette zur Metadaten-Extraktion. Das nachfolgende Kapitel stellt ein Workflow-System für Digitalisierungsprozesse vor, welches Metadaten als integralen Bestandteil eines Digitalisats betrachtet und durch automatisierte Extraktions-Module vielfältige Vorteile während der Produktion und in der Nachnutzung erfährt.

9 Workflow-System für die Retro-Digitalisierung

In dieser Arbeit wurde ein Verfahren vorgestellt, mit dessen Hilfe der Kontext eines Terms zu dessen Klassifikation in syntaktisch schwierigem Terrain und bei häufig wechselnder Domäne verwendet werden kann. Die vorherigen Kapitel haben illustriert, wie dieses Verfahren in Gestalt eines Klassifikationsframeworks in die Praxis umgesetzt werden konnte und dessen Effektivität belegt. Nachfolgend soll nun am Beispiel eines Workflow-Systems für Digitalisierungsprozesse dargestellt werden, wie das Klassifikationsframework in eine Verarbeitungskette zur Digitalisierung von Dokumenten eingegliedert werden kann und so strukturelle und semantische Metadaten zur weiteren Nachnutzung zur Verfügung stellen kann. Dabei soll besonders herausgearbeitet werden, wie in Abgrenzung zu konkurrierenden Workflow-Systemen die Verwertung der gesammelten Metadaten bereits im Digitalisierungs-Workflow zur Steigerung der Produktions-Effektivität und -Qualität beitragen sowie dem Nutzer eine originalgetreuere und intuitivere Darstellung des Quelldokuments ermöglichen.

Das vorgestellte Workflow-System ist in Kooperation mit der Digitalisierungsabteilung der Universitätsbibliothek Würzburg entstanden und stellt eine Software-Lösung dar, welche den Digitalisierungsprozess unterstützt und eine computergestützte Metadaten-Anreicherung selbst bei schwierigem Ausgangsmaterial ermöglicht. Es stellt einen Ansatz dar, der die in Kapitel 1.1 ausführlich beschriebenen Ziele, Digitalisate einem breiten Publikum online zur Verfügung zu stellen und die Werke langfristig zu konservieren, aus einem System ganzheitlich unterstützt.

Dieses Kapitel stellt die Vorteile eines Workflow-Systems, welches sich der originalgetreuen Digitalisierung eines Werks unter Berücksichtigung werksspezifischer Metadaten verschrieben hat, gegenüber gängigen, auf Masse und Geschwindigkeit ausgelegte Verfahren vor und zieht einen Vergleich zu anderen, konkurrierenden Workflow-Systemen.
Der Gewinn, den man durch Einsatz des Workflow-Systems erhält, soll schlussendlich nicht nur anhand einer reduzierten Verarbeitungszeit und verbesserter Produktionsqualität, sondern auch anhand des Mehrwerts zusätzlich verfügbarer, automatisiert gewonnener Metadaten belegt werden, die zu einer detaillierteren und originalgetreueren Digitalisierung des Originals führen. Das nachfolgende Kapitel gibt daher Aufschluss über die Motivation hinter dem Workflow-System, geht auf Besonderheiten bei der Implementierung ein und stellt an Beispielen die Effektivität sowie die Alleinstellungsmerkmale vor.

9.1 Problembeschreibung

Viele der Objekte, die den Retro-Digitalisierungsprozess (wie definiert auf Seite 22) im Digitalisierungszentrum der Universitätsbibliothek Würzburg durchlaufen, sollen auf Online-Plattformen [Würzburg (2014a)], [Würzburg (2014b)] einem breiten Publikum zur Verfügung gestellt werden. Um den Besuchern ein möglichst detailgetreues und umfassendes Bild des Objekts zu liefern, werden neben den bloßen Bilddaten auch bibliographische Metadaten, Übersetzungen, Annotationen, Erläuterungen, Volltexte und Querverweise zu anderen Dokumenten angeboten (siehe Beispiel *Epistolae Pauli* [Apostulus (790)]). Die Erfassung und Pflege dieser Daten ist aufgrund der Domäne (wie in Kapitel 1.1 dargelegt) üblicherweise ein zeitaufwändiger, rein manueller Vorgang, der das Eingreifen eines Experten unumgänglich macht.

Mit steigendem Produktionsvolumen wurde es für die Mitarbeiter jedoch zunehmend schwieriger, Ressourcen, Zeitpläne und Personal zu koordinieren. Die komplexen Anforderungen eines prestigeträchtigen Digitalisierungsprojekts (*Libri Sanc-*

ti Kiliani [Würzburg (2014a)]) machten es schließlich unumgänglich, die bisher rein manuelle Verarbeitungskette der Digitalisierungsabteilung mit einem Workflow-System zu unterstützen. Einige der zu digitalisierenden Werke besitzen (abgesehen von ihrer kulturellen Bedeutung) einen substanziellen materiellen Wert. Für die Digitalisierung dieser wertvollen und äußerst sensiblen Werke werden eigens für diesen Anwendungsfall konstruierte, komplexe bildgebende Verfahren angewandt, die neben höchster Reproduktions-Qualität vor allen Dingen einen schonenden Umgang mit den Quellen anstreben. Das Konzept, welches dem Workflow-System zugrunde liegt, wurde in der Entwurfsphase speziell auf die Anforderungen der Digitalisierungsabteilung zugeschnitten. Inbesondere soll mit Hilfe des Workflow-Systems der Anteil menschlicher Beteiligung am Digitalisierungsprozess deutlich reduziert werden. Einige der zentralen Anforderungen an das System sollen nun in den nachfolgenden Abschnitten erläutert werden.

9.2 Zielbeschreibung

Um die Leistung eines Workflow-Systems bemessen zu können, muss zunächst der Begriff der *Durchlaufzeit* eingeführt werden. Das Wirtschaftslexikon [*wirtschaftslexikon24.com* (2014)] definiert diesen wie folgt:

> „Die Durchlaufzeit ist die Zeitspanne zwischen der ersten Materialentnahme bei der Fertigung und der Übergabe des Erzeugnisses an den Auftraggeber oder das Absatzlager. Die Durchlaufzeit ist die Gesamtdauer für die Erledigung einer Aufgabe, in einer Organisation: Summe aus Bearbeitungszeit(en), Transportzeit(en) und Liegezeit(en).“

Die Durchlaufzeit eines Digitalisats umfasst neben der Bilddaten-Produktion auch die Metadaten-Erschließung und -Aufbereitung. Die Leistungsfähigkeit eines Workflow-Systems sollte sich also direkt an einer Veränderung der Durchlaufzeit ablesen lassen. Ein Ziel ist eine eindeutige Verbesserung der Durchlaufzeit gegenüber der Produktionskette, die nicht durch ein Workflow-System gestützt wird. Weiterhin

fließen in die Leistungsbewertung zusätzliche, automatisiert gewonnene strukturelle und bibliographische Metadaten ein, die ohne das spezialisierte Workflow-System nicht erfasst werden könnten.

Reduktion der Durchlaufzeit

Um die Durchlaufzeit eines Digitalisierungsprojektes reduzieren zu können, müssen zunächst diejenigen Faktoren identifiziert werden, die negative Auswirkungen auf die Durchlaufzeit haben. Die *Fehlerrate* einer Produktionskette gehört dabei offensichtlich zu den bedeutendsten Faktoren:
Je nach Tragweite eines Fehlers müssen einzelne Arbeitsschritte, schlimmstenfalls sogar der gesamte Prozess erneut abgewickelt werden, mit direktem Effekt auf die Durchlaufzeit. Abbildung 9.1 zeigt ein typisches Manuskript, wie es im Digitalisierungszentrum verarbeitet wird.

Abbildung 9.1: Typisches Manuskript mit beschädigtem Einband.

Auffällig ist der schlechte Erhaltungszustand des Einbands und des Buchrückens. Muss nun dieses Dokument aufgrund eines zu spät bemerkten Produktionsfehlers

erneut digitalisiert werden, riskiert man eine zusätzliche Beschädigung der sensiblen Quelle. Die frühzeitige und zuverlässige Fehlererkennung ist somit eine Kernanforderung an das Digitalisierungs-System, welche die Entwurfsphase geprägt hat. Die Ursachen für Fehler im Digitalisierungs-Workflow können dabei unterschiedlichen Kategorien zugeordnet werden:

Menschliche Beteiligung

Zum einen werden die Geräte von einem kleinen Kern fester Mitarbeiter und einem größeren, häufig wechselndem Kreis fachlich wenig oder gar nicht speziell ausgebildeter Hilfskräfte (zumeist Studenten) bedient. Es obliegt dem ausgebildeten Fachpersonal, wichtige Schritte während des Digitalisierungsvorgangs zu kontrollieren und gegebenenfalls eine Nachbesserung anzufordern. Die Sichtung des Materials stellt hohe Anforderungen an die Konzentration der Mitarbeiter: Da ein typisches Digitalisierungsprojekt üblicherweise mehrere hundert Aufnahmen in hoher Auflösung umfasst und Bildfehler oft von subtiler Natur sind, werden diese bei stichprobenartiger Kontrolle oder flüchtiger Ansicht schnell übersehen.

Fehlerträchtige Ressourcen

Weiterhin durchlaufen die Bilddaten eines Digitalisierungsprojekts mehrere Arbeitsschritte, welche unter Umständen von verschiedenen Mitarbeitern an verschiedenen Rechnern durchgeführt werden. Bei den dafür notwendigen Migrations-Vorgängen der Daten ist nicht nur der Mensch eine Fehlerquelle (Kopieren an falsche Stelle, Kopieren einer veralteten Version), auch die dazu verwendeten Ressourcen sind inhärent fehlerträchtig, als Beispiele sei der Abbruch einer Netzwerk-Verbindung mit resultierendem Datenverlust genannt. Die Kooperation der Mitarbeiter über verschiedene Betriebssysteme hinweg (MacOS, Linux, Windows) birgt in der Praxis Spielraum für weitere Schwierigkeiten.

Domäne

Wie in Kapitel 1.1 beschrieben, ist neben den angesprochenen Fehlern auch die Domäne selbst fehlerträchtig. Eine automatisierte Text-Extraktion auf Werken wie in Abbildung 9.1 mit Optical Character Recognition (OCR) Ansätzen schlägt zumeist wegen des schwierigen Schriftbilds oder des Erhaltungszustands des Werks fehl. Ebenfalls scheitern gängige Verfahren zur unterstützenden Metadaten-Extraktion wie Named Entity Recognition oder Wörterbuch-Analysen häufig an den Besonderheiten der Domäne, was das Eingreifen eines menschlichen Experten notwendig werden lässt. Module, die im Rahmen des Workflow-Systems eine Metadaten-Anreicherung unterstützen sollen, müssen daher speziell an die Eigenschaften der Domäne angepasst werden.

Das Workflow-System soll also die Durchlaufzeit für Digitalisierungsprojekte reduzieren, indem der Faktor mit dem größten Einfluss auf die Durchlaufzeit, der Produktionsfehler, minimiert wird. Dazu müssen Fehler in der Produktionskette frühzeitig erkannt, Programmfehler robust und konsistent behandelt und die Verarbeitungskette effizient implementiert werden.

Neben der Reduktion der Durchlaufzeit haben noch weitere Zielsetzungen den Entwurf des Workflow-Systems geprägt:

Variable Produktionsketten

Die Digitalisierungsabteilung bietet neben ihrer Spezialisierung auf alte Handschriften, Landkarten und historische Drucke auch Auftragsarbeiten an, die von der Reproduktion zeitgenössischer Werke bis hin zu Sonderwünschen wie der Digitalisierung von kompletten Schulwandtafeln alles beinhalten kann. Je nach digitalisiertem Objekt und dessen Verwendungszweck unterscheiden sich die einzelnen Verarbeitungsschritte dabei von Projekt zu Projekt deutlich. Wie in Abschnitt 9.4 beschrieben, bietet das Workflow-System eine breite Palette von Analyse-, Export-

und Metadaten-Extraktions-Modulen an; allerdings ist nicht jedes Modul für jedes Digitalisat und jeden Verwendungszweck konzipiert: Die Digitalisierung und Metadaten-Anreicherung erfordert bei einer historischen Landkarte andere Verarbeitungswege als bei einer zeitgenössischen Handschrift. Das Workflow-System muss daher in der Lage sein, die Produktionsschritte dynamisch in Abhängigkeit vom jeweiligen Digitalisat und dessen Verwendung anzupassen.

Metadaten-Erfassung

Neben den reinen Bilddaten sollen möglichst viele Aspekte des originalen Dokuments erfasst werden. Das Workflow-System soll mit automatisierten Ansätzen eine Metadaten-Erfassung erleichtern und eine nachträgliche Pflege der erfassten Daten ermöglichen. Die automatisierten Ansätze müssen mit den Besonderheiten der Domäne umzugehen wissen. Die gewonnenen Metadaten werden eingesetzt, um in Form von Fehlerheuristiken Produktionsfehler frühzeitig zu erkennen und erlauben originalgetreuere und intuitivere Präsentationstechniken.

Präsentation und Langzeit-Archivierung

Kapitel 1.1 stellt neben konservatorischen Aspekten die Zugänglichkeit für ein breites Publikum als Hauptgründe für die Digitalisierung von Dokumenten vor. Ein Workflow-System für Digitalisierungsprozesse muss diese Ziele modular unterstützen, indem Plattformen zur Langzeit-Archivierung bedient werden und Präsentations-Systeme beschickt werden können. Bei nachträglicher Metadaten-Pflege müssen die Änderungen in die Ziel-Plattformen übernommen werden. Die Anbindung an Lösungen zur Langzeit-Archivierung soll eine sichere Verwahrung des Digitalisats gewährleisten.

Zusammenfassung

Erklärtes Entwurfsziel des Workflow-Systems ist eine Minimierung der Durchlaufzeit eines Digitalisierungs-Auftrags zur effizienten und schonenden Verarbeitung von sensiblen Quellen. Es ist daher von zentraler Wichtigkeit, mit den vielfältigen bekannten Fehlertypen umgehen zu können, aber auch im Falle von bisher unberücksichtigten Fehlern in einem konsistenten System-Zustand verbleiben zu können. Wenn möglich, sollen Fehler automatisiert behoben, zumindest aber frühzeitig erkannt und einem Mitarbeiter zur Inspektion vorgelegt werden. In Abgrenzung zu auf dem Markt verfügbaren Lösungen (siehe nachfolgendes Kapitel) wird vom Workflow-System eine **zuverlässige Fehlererkennung und hohe Fehlertoleranz** erwartet, um das Risiko von wiederholten Digitalisierungen zu minimieren und gleichzeitig höchste Reproduktionsqualität zu gewährleisten. Dieses Ziel steht im deutlichen Gegensatz zu auf Geschwindigkeit ausgelegten Massendigitalisierungs-Verfahren (siehe Abschnitt 9.3).
Eine **automatisierte Unterstützung zur Metadaten-Erfassung und -Pflege** soll die Notwendigkeit menschlichen Eingreifens reduzieren und neben den rein visuellen Daten auch werksspezifische Metadaten liefern. Insbesondere sollen dabei die kontext-basierten Techniken, die im Kapitel 5 vorgestellt wurden, modular und erweiterbar eingebunden werden. Die Metadaten werden als Fehlerheuristiken eingesetzt, um Produktionsfehler frühzeitig zu erkennen und sollen in spezialisierten Präsentationstools Mehrwerte gegenüber den bloßen Bilddaten liefern.
Das System muss darüber hinaus in der Lage sein, in Abhängigkeit vom Medium und dessen Verwendungszweck **dynamische Ausführungspfade** zu wählen. Nach erfolgreicher Bearbeitung im Workflow-System soll das Digitalisat an eine **Langzeit-Archivierungslösung** übergeben werden können, aber auch auf **Online-Präsentations-Portalen** einem breiten Publikum möglichst originalgetreu zugänglich gemacht werden. Nachträgliche Änderungen und Pflegearbeiten am Digitalisat und den Metadaten sollen möglich sein.

9.3 Abgrenzung zu existierenden Workflow-Systemen

Es gibt viele kommerzielle und freie Workflow-Systeme, die allgemeine Komponenten zur Unterstützung bei Projektmanagement sowie Zeit- und Ressourcen-Planung bieten, sowie auch speziell auf die Digitalisierung zugeschnittene Lösungen:

Goobi

Goobi[1] ist ein Projekt aus dem Bereich der Workflow-Systeme, die speziell auf die Problemstellungen rund um Digitalisierungen zugeschnitten sind. Wie auch das im Digitalisierungszentrum der Universität Würzburg eingesetzte System ist es eine plattformunabhängige, modulare Anwendung, deren Komponenten für unterschiedliche Aufgaben im Digitalisierungs-Workflow verantwortlich sind. Während sich Goobi allerdings zum Ziel setzt, das Projekt- und Ressourcen-Management zu erleichtern und damit verbundene Probleme zu lösen, geht das Workflow-System des Würzburger Digitalisierungszentrums darüber hinaus und bietet automatisierte Module, die Digitalisate auf Produktionsfehler analysieren und damit die Fehlerrate deutlich senken können. Zur effizienten Bewältigung der dafür notwendigen Analysen werden Rechner im Netzwerk hinzugezogen. Die Qualitätskontrolle obliegt in Goobi hauptsächlich menschlichen Mitarbeitern und ist damit fehleranfällig. Weiterhin fehlen in Goobi Module zur automatisierten Metadaten-Extraktion. Ebenfalls bietet Goobi derzeit keine Anbindung an Lösungen zur Langzeit-Archivierung.

[1] http://www.goobi.org

ZEND

Ein anderes Produkt ist ZEND[2] des Münchner Digitalisierungszentrums: Das modulare Workflow-System bietet Möglichkeiten, ein Digitalisat mit Metadaten anzureichern - allerdings geschieht das auf rein manuellem Wege. Das in diesem Kapitel vorgestellte Workflow-System bietet hingegen automatisierte Module zur Metadaten-Extraktion, die Information-Retrieval-Verfahren auf die Digitalisate anwenden können und so den Anteil manuellen Eingreifens reduzieren. Automatisierte Qualitäts-Kontrollen fehlen ebenso. ZEND besitzt Visualisierungs-Werkzeuge, um die Digitalisate anzuzeigen, jedoch fehlt die Vielfalt der Präsentations-Werkzeuge, die aus dem vorgestellten Workflow-System beschickt werden können. Eine synoptische Visualisierung von Bild- und Metadaten ist nicht vorhanden.

Kepler

Darüber hinaus gibt es eine Vielzahl von Workflow-Systemen, mit deren Hilfe sich abstrakte Arbeitsabläufe modellieren und Mitarbeiter sowie Ressourcen dezentral koordinieren lassen (beispielsweise Kepler[3]). Allerdings lassen diese Systeme die auf den Anwendungsfall spezialisierten Werkzeuge vermissen, die letztendlich dazu führen, dass Arbeitsabläufe beschleunigt werden. Durch den Einsatz eines solchen Workflow-Systems würde sich der Anteil manuellen Eingreifens in die Arbeitsabläufe also nicht verringern. Weitere Workflow-Systeme werden in [Schöneberg, Schmidt und Höhn (2013)] besprochen.

Google Books

An dieser Stelle sollen auch große Massendigitalisierungs-Initiativen wie Google Books nicht unerwähnt bleiben: Einigen Quellen zufolge hat Google inzwischen

[2] http://www.digitale-sammlungen.de/index.html?-c=digitalisierung-zend
[3] https://kepler-project.org/

trotz nicht vollständig geklärter urheberrechtlicher Fragen bereits 20 Millionen Bücher digitalisiert (Stand des Berichts [*LePublikateur* (2013)] im November 2013). Diese beeindruckende Zahl steht jedoch im Gegensatz zur scharfen Kritik einiger Experten, welche die Qualität der Digitalisate beklagen. In [*Frankfurter Rundschau* (2013)] wird beispielsweise beschrieben, dass die Finger von Mitarbeitern auf der digitalen Kopie zu erkennen seien, die Werke teilweise falsch kategorisiert und durch werksunspezifische Digitalisierung sogar schlicht unbrauchbar gemacht wurden. Abbildung 1.2 auf Seite 15 zeigt beispielhaft einen grob digitalisierten Ausschnitt aus „Die Leiden des jungen Werther"von Johann Wolfang von Goethe. Google zielt laut eigener Aussage[4] darauf ab, den Bestand durchsuchbar und einem breiten Publikum zugänglich zu machen. Auffällig ist jedoch die eher grobe Natur der präsentierten Werke (zumeist binarisierte Bilder). Volltexte, die ein wissenschaftliches Arbeiten ermöglichen, fehlen oft. Weiterhin sind große, nach einem Algorithmus zuvor berechnete Teile urheberrechtlich geschützter Werke dem Nutzer vorenthalten (vgl.[*LePublikateur* (2013)]).
Das vorgestellte Workflow-System soll die Abgrenzung zur Massendigitalisierung deutlich machen, indem es Schwerpunkte auf höchste Qualität der präsentierten Bilddaten legt, möglichst viele Aspekte des Originals zu erfassen versucht, Metadaten (automatisiert) extrahiert und dem Nutzer einen authentischen Eindruck des Dokuments ermöglicht.

Besonderes Alleinstellungsmerkmal des Workflow-Systems ist die ganzheitliche Verantwortung für alle Aspekte des Dokumenten-Lebenszyklus: Nachdem ein digitalisiertes Werk die komplette Verarbeitungskette durchlaufen hat, endet die Zuständigkeit des Workflow-Systems nicht. Vielmehr sorgt es für die Übergabe des Digitalisats an eine Langzeit-Archivierungslösung, stellt Werkzeuge zur nachträglichen Pflege von werksspezifischen Metadaten zur Verfügung und erlaubt per Knopfdruck eine Veröffentlichung im Netz. Die dort verwendeten Präsentations-Tools machen von den gesammelten strukturellen und bibliographischen Metadaten Gebrauch und bieten dem Besucher optische und inhaltliche Mehrwerte sowie einen originalgetreueren Eindruck des Dokuments. Am Ende dieses Kapitels finden

[4] http://books.google.com/

sich einige Beispiele veröffentlichter Werke.

9.4 Implementierung

Dieser Abschnitt illustriert das Konzept des Workflow-Systems, einige Implementierungs-Details und die Umsetzung der zuvor genannten Entwurfsziele.

Konzept

Das Workflow-System begleitet ein Digitalisierungs-Projekt von der Auftragsannahme bis zur Archivierung (siehe Definition „Durchlaufzeit“ in Abschnitt 9.2). Abbildung 9.2 zeigt die wichtigsten Stationen während der Bearbeitung: Nachdem ein Digitalisierungs-Projekt für die Buchhaltung erfasst wurde, müssen zunächst die Bilddaten erzeugt werden. Sobald diese vorhanden sind, befindet sich das Projekt in einer Analyse-Feedback-Schleife. Diese wird solange durchlaufen, bis sämtliche Qualitätsziele erfüllt sind. Innerhalb dieser Schleife versuchen automatisierte Module, gängige Fehlertypen zu erkennen sowie eine Metadaten-Extraktion vorzunehmen und legen die Resultate dem Anwender zur Begutachtung vor. Ein qualitätsgeprüftes Projekt kann danach für die Langzeitarchivierung oder den Export zu Präsentationsplattformen oder Metadaten-Portalen vorbereitet werden.

Zur Umsetzung der gesetzten Ziele wurde die Philosophie des *Reaktiven Manifests* (*The Reactive Manifesto* [*The Reactive Manifesto* (2014)]) berücksichtigt: Laut diesem sollte ein System idealerweise *event-driven*, *robust*, *skalierbar* und *reaktionsfähig* sein. Abbildung 9.3 verdeutlicht das Konzept. Nachfolgend soll nun gezeigt werden, wie durch die Umsetzung der Richtlinien im Reaktiven Manifest die zu Beginn des Kapitels vorgestellten Entwurfsziele erreicht werden konnten:

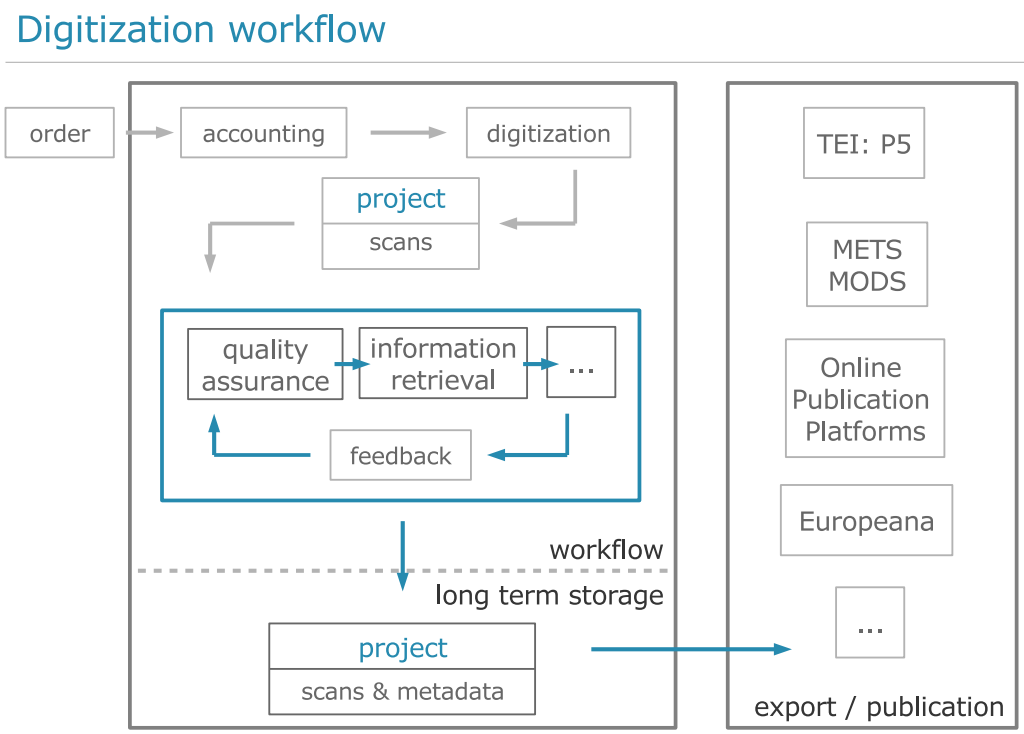


Abbildung 9.2: Allgemeiner Aufbau des Programms.

Event-driven Design

Das Workflow-System ist in eine leichtgewichtige Client- und eine Server-Komponente aufgeteilt. Client und Server kommunizieren miteinander über eine vollständig auf *Events* basierende, asynchrone Schnittstelle und verwenden dabei simple Text-Nachrichten (*event-driven design*). Nach [*The Reactive Manifesto* (2014)] besitzt ein System, welches einem auf Events basierenden Entwurf folgt, folgende Vorteile: Die miteinander kommunizierenden Komponenten sind nur lose gekoppelt, da ihre Schnittstelle lediglich die ausgetauschten Text-Nachrichten sind. Alle Komponenten können somit unabhängig voneinander entwickelt werden. Durch die asynchrone Nachrichtenverarbeitung werden darüber hinaus die Systemressourcen effizienter genutzt, was einen höheren Durchsatz und eine geringere Latenz mit sich bringt.

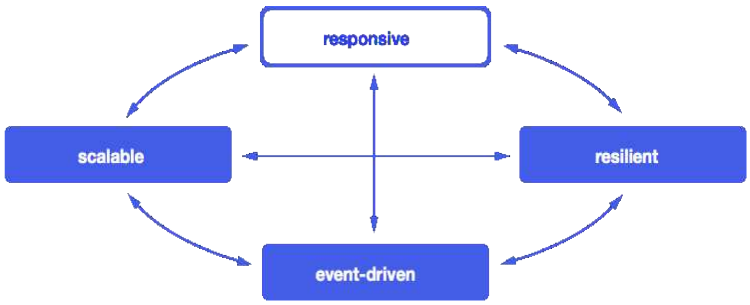


Abbildung 9.3: Eigenschaften eines reaktiven Systems. Quelle: [*The Reactive Manifesto* (2014)].

Für das Versenden von Nachrichten existieren bereits vielfältige erprobte Lösungen. Analog zu Entwurfsmustern im klassischen Software-Entwurf finden sich in [Hohpe und Woolf (2003)] viele Entwurfsmuster - so genannte Enterprise Integration Patterns (EIP)[5] - für Szenarien rund um das Versenden von Nachrichten. Innerhalb des Workflow-Systems wird von der Nachrichtenebene mit Hilfe von Apache Camel[6] abstrahiert, einem Framework, welches die meisten der EIPs implementiert, was ein komplexes und dynamisches Versenden von Nachrichten zwischen den Programmteilen erlaubt.

Abbildung 9.4 veranschaulicht, wie mit Hilfe des EIP: Pipes and Filters[7] zur Laufzeit genau die Komponenten zu einer dynamischen Verarbeitungskette zusammengesetzt werden können, die für den Typ des vorliegenden Digitalisats und dessen Verwendungszweck passend sind.

Robustheit

Den schwerwiegendsten Einfluss auf die Durchlaufzeit eines Digitalisierungsprojekts haben Produktionsfehler: Wie eingangs erwähnt, wirken sich unbemerkte

[5] http://www.eaipatterns.com/eaipatterns.html
[6] http://camel.apache.org/
[7] http://www.eaipatterns.com/PipesAndFilters.html

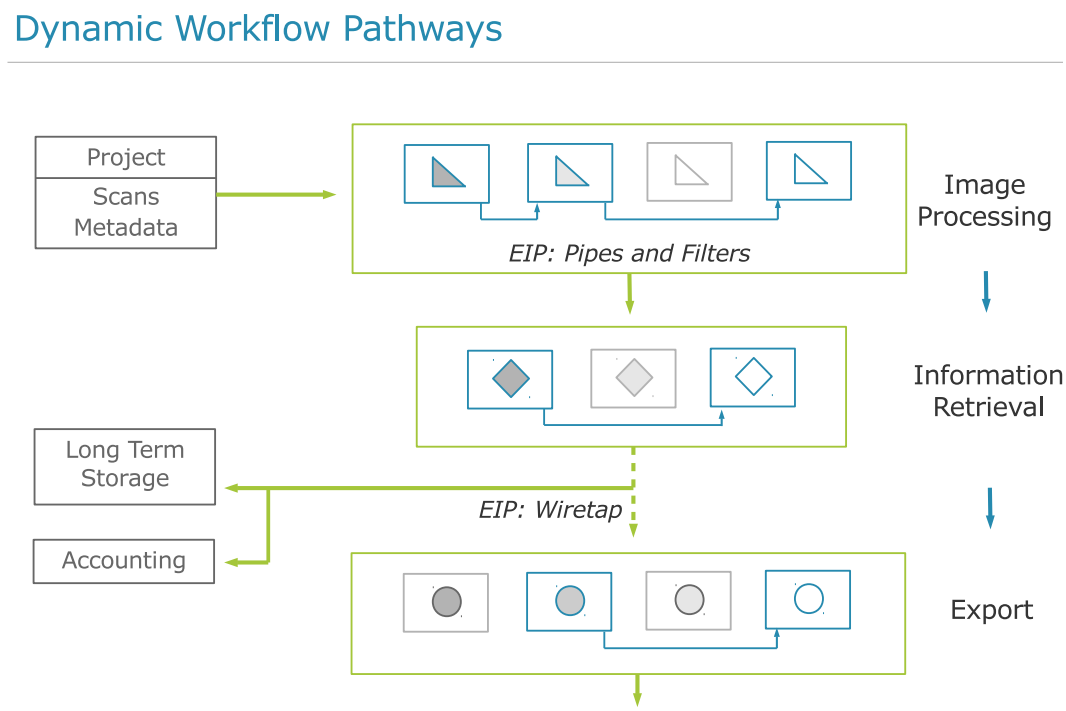


Abbildung 9.4: Dynamische Workflow-Routen.

Produktionsfehler negativ auf die Qualität des Digitalisats aus, erkannte Fehler müssen hingegen zeitaufwändig behoben werden. Während der Entwicklung des Workflow-Systems stellte sich schnell heraus, dass es unmöglich ist, bereits im Vorfeld alle möglichen (und zukünftig auftretenden) Fehlertypen zu erfassen und entsprechende Gegenmaßnahmen bereitzuhalten. Der Einsatz in einer inhärent fehlerträchtigen Domäne (I/O-Operationen über das Netzwerk, über Datenträger, menschliche Beteiligung) stellt hohe Anforderungen an die Fehlertoleranz und Robustheit des Systems. Laut [*The Reactive Manifesto* (2014)] ist die Robustheit einer Anwendung essenziell, um Ausfallzeiten (und damit die Durchlaufzeit) zu minimieren. Daher wurde eine alternative Strategie gewählt: Anstatt im Vorfeld zu versuchen, sich auf jeden (auch zukünftigen) Fehlertyp vorzubereiten, wurde das System so entworfen, dass es auch mit unbekannten Fehlerzuständen umgehen kann.

Bei der Umsetzung dieses Ansatzes wurde auf das Aktoren-Modell gesetzt, wie es von Akka bereitgestellt wird. Vorteilhaft war insbesondere die hierarchische Natur der Aktor-Systeme: Wie in Kapitel 6.2 beschrieben, hat jeder Aktor einen übergeordneten Eltern-Aktor, der über Veränderungen im Lebenszyklus seiner Kind-Aktoren sowie über auftretende Fehler informiert wird.
Das Workflow-System wurde daher so entworfen, dass kritische oder fehlerträchtige Abschnitte stets von Aktoren abgewickelt werden, die als Blatt in der Aktor-Hierarchie angeordnet sind. Da diese stets von einem Eltern-Knoten überwacht werden, der im Fehlerfall regulierend eingreifen kann, beschränken sich Anwendungsfehler stets auf den ausführenden Aktor und beeinflussen andere Programm-Abläufe nicht. Durch diesen Entwurf bleibt die Anwendung selbst im Fehlerfall in einem konsistenten Zustand und erfüllt die eingangs formulierte Anforderung an einen robusten Anwendungs-Entwurf.

Dieses Konzept wurde mit Hilfe einer modifizierten Version des *Master-Worker*-Entwurfsmusters (wie beschrieben in [Wyatt (2013)]) umgesetzt. In [Tiona (2013)] werden unterschiedliche Implementierungen dieses Musters entwickelt und bewertet: Es definiert einen Aktor (*master*), der eingehende Arbeitspakete, ihren Bearbeitungsstatus und ihren jeweiligen Bearbeiter (*worker*) verwalten kann. Der *master* bearbeitet dabei selbst keine der eingehenden Arbeitspakete, sondern delegiert diese an *worker*, sofern vorhanden. Er ist insbesondere auch nicht für die Erzeugung von *worker*-Aktoren verantwortlich, sondern erlaubt diesen nur, sich bei ihm zu registrieren und nach Arbeitspaketen zu verlangen. Abbildung 9.10 illustriert das Konzept. Code-Beispiel 9.1 zeigt einen Auszug aus dem *master*-Aktor (in Scala), der die regenerativen Fähigkeiten des Ansatzes belegt: Zeile 15 definiert, dass dem *master* zugeordnete Kind-Aktoren nach Auftreten eines Fehlers (Exception) stets neu-gestartet werden sollen. In Zeile 66 wird jeder *worker*, der sich am *master* registriert, aktiv überwacht. Sollte nun ein *worker* beispielsweise aufgrund eines Netzwerk-Fehlers nicht mehr erreichbar sein, erhält der *master* eine entsprechende Nachricht und verteilt die vom nicht mehr erreichbaren *worker* bearbeitete Aufgabe anderweitig (Zeilen 24 und 30ff).

Diese topologie-agnostische, pull-basierte Implementierung spielt eine wichtige Rolle für die Skalierbarkeit der Anwendung und soll später ausführlicher erläutert werden. Der *master*-Worker kann explizite Ausweich-Strategien für bekannte Fehlertypen zur Verfügung stellen, sollten *worker*-Aktoren bei der Erledigung ihrer Arbeiten scheitern. Die resultierende Struktur ist (selbst für im Vorfeld nicht berücksichtigte Fehlerfälle) äußerst robust und kann sich selbst reparieren.
Durch die Implementierung des Master-Worker-Musters verblieb das System selbst bei schwerwiegenden Fehlern wie dem Ausfall einer (Netzwerk-) Ressource in einem konsistenten Zustand. Ausführliche Ergebnisse werden in Abschnitt 9.4 auf Seite 135 demonstriert.

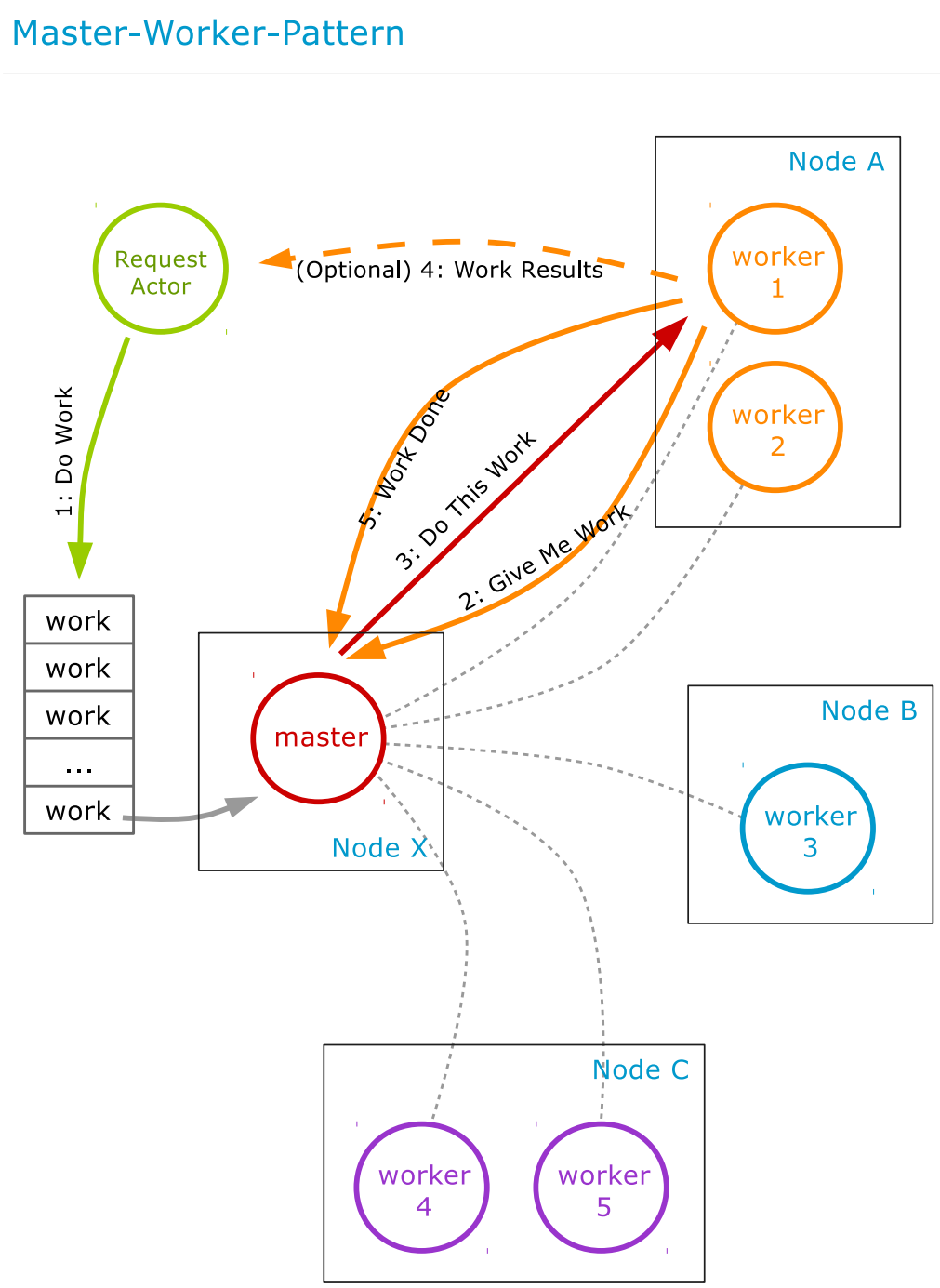


Abbildung 9.5: Master Worker Pattern. Basierend auf: [Wyatt (2013)]

```
0  class Master extends Actor with ActorLogging {
1    import context._
2
3    // an abstract work item and its owner
4    type Workload = (ActorRef, Any)
5
6    // -- actor state:
7    // workers and their workloads
8    var workers = Map.empty[ActorRef, Option[Workload]]
9
10   override def preStart() = {
11     log.debug("Master is starting.")
12   }
13
14   // in case of any error, restart the worker
15   override val supervisorStrategy = OneForOneStrategy() {
16     case _ => Restart
17   }
18
19   /** processes incoming messages and keeps track of pending work */
20   def receive = {
21     // an actor was created
22     case WorkerCreated(worker) => workerCreated(worker)
23     // a worker terminated
24     case Terminated(worker) => workerTerminated(worker)
25     ...
26   }
27
28
29   /** after a worker died, its work re-dispatched */
30   private def workerTerminated(worker: ActorRef) = {
31     workers.get(worker) map {
32       case Some((owner, workload)) => {
33         log.error("A busy worker died: {}. Rescheduling his work.", worker)
34         // reschedule work
35         self.tell(workload, owner)
36         // remove worker
37         workers -= worker
38         // cancel timeout checks
39         pending.get(worker) map (_.cancel)
40         pending -= worker
41       }
42       case None => log.error("An idling worker died: {}.", worker)
43     }
44
45     workers -= worker
46   }
47
48
49   /** will be called upon worker creation */
50   private def workerCreated(worker: ActorRef) = {
51
52     // if we already know this worker, it might have restarted
53     workers.get(worker) map {
54
55       // reschedule the work
56       case Some((owner, workload)) => {
57         self.tell(workload, owner)
58         pending.get(worker) map (_.cancel)
59         pending -= worker
60       }
61
62       // add worker as idle
63       case None => {
64         log.info("A worker has been created: {}.", worker)
65         // monitor worker
66         context watch worker
67       }
68     }
69
70     // ... and add to map of workers
71     workers += (worker -> None)
72
73     // broadcast work, if any
74     notifyWorkers()
75   }
76
77 ...
```

Listing 9.1: Master-Aktor des master-worker-Musters

Automatisierte Qualitätskontrollen

Das erklärte Entwurfsziel ist es, die Produktionsfehlerrate (und damit die Durchlaufzeit) zu minimieren. Mit Hilfe der Aktor-Hierarchien kann das Workflow-System zwar effektiv gegen vielfältige Fehlerszenarien und inkonsistente Zustände abgesichert werden, Produktionsfehler im Digitalisierungsworkflow werden dadurch allerdings noch nicht gefunden und die Gefahr der erneuten Digitalisierung einer empfindlichen Quelle ist damit noch nicht unterbunden.
Zu diesem Zweck wurde das Workflow-System mit einer breiten, leicht erweiterbaren Palette von Analyse-Modulen versehen, die die eingehenden Bild- und Metadaten automatisiert auf häufige Fehlertypen untersuchen können:

Produktionsfehler Die Module zur frühzeitigen Fehlererkennung in den Bilddaten werden über eine schmale Schnittstelle *AnalyseModul* eingebunden (siehe Code-Beispiel 9.2): Ein AnalyseModul erwartet als Eingabe das zu analysierende Bild und dessen Quelldatei. Nachfolgend werden einige konkrete Implementierungen dieser Schnittstelle beschrieben. Entwickler können leicht eigene Module zur Fehlererkennung in die Verarbeitungskette eingliedern: Dazu muss lediglich das oben beschriebene Interface implementiert und im Classpath zur Verfügung gestellt werden. Das Modul wird dann zur Laufzeit über den *ServiceLoader*[8] Mechanismus gefunden.

Erkennung von Unschärfe Ein häufiger Fehler, der bei der Digitalisierung auftreten kann, ist eine partielle oder vollständige Unschärfe eines Bildes durch fehlerhafte Einstellungen am bildgebenden Gerät oder Bedienungsfehler (beispielsweise zu frühes Entfernen des Dokuments bei noch nicht abgeschlossenem Scan-Vorgang). Das Unschärfe-Modul ermittelt mit Hilfe von Hochpass-Filtern (zB. Sobel-Operator) Kanten innerhalb des Bildes und kann bei abweichenden Kanten-Verläufen in verschiedenen Bildabschnitten auf unscharfe Bereiche schließen. Abbildung 9.6 zeigt die erfolgreiche Erkennung eines unscharfen Bereichs innerhalb eines Beispielbildes.

[8] http://docs.oracle.com/javase/7/docs/api/java/util/ServiceLoader.html

Erkennung von Ausrichtungsfehlern Durch falsche Positionierung des zu digialisierenden Gegenstands kann es zu einer gedrehten Aufnahme kommen. Um solche Fehlerfälle zu finden, analysiert das Drehfehler-Modul den Abstand zwischen dem Rand des abgebildeten Dokuments und dem Rand der Aufnahme auf Unregelmäßigkeiten. Dazu werden über Hochpassfilter zunächst Kanten im Bild detektiert und ihr Winkel zum Bildrand vermessen.

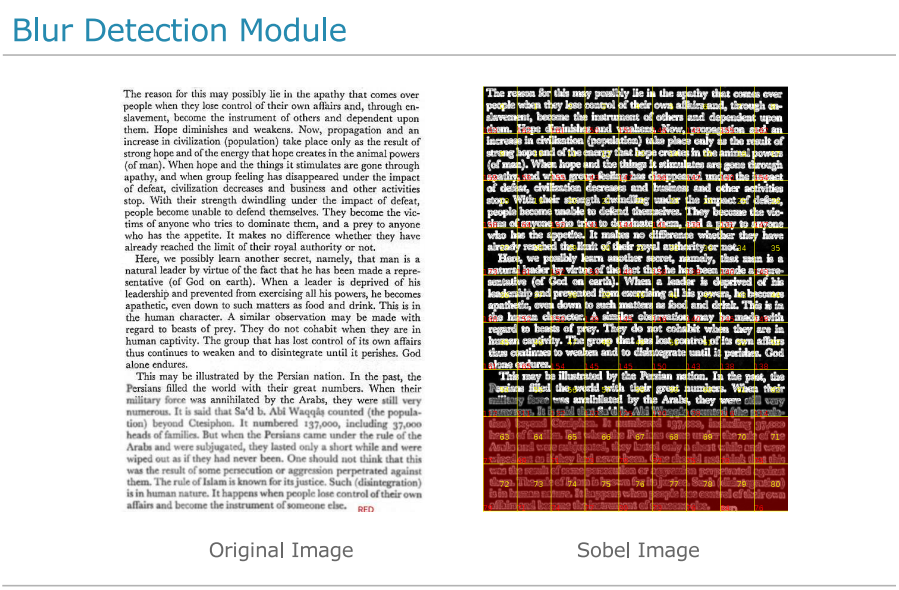


Abbildung 9.6: Analyse-Modul zur Erkennung von Unschärfe.

Objekt-Erkennung Für einige Aufnahmen kann es notwendig sein, das Dokument mit Halterungen oder Klammern zu fixieren. Mit Hilfe eines Analyse-Modules, welches diese Halterungen zu erkennen vermag, soll versucht werden, diese aus dem Bild zu retuschieren. Abbildung 9.7 zeigt ein Bild nach erfolgreicher Anwendung des Moduls zur Objekt-Erkennung. Ein verwandtes Modul erlaubt die Unterscheidung zwischen der digitalisierten Seite des Dokuments und dem Hintergrund der Aufnahme und ermöglicht eine automatisierte Erstellung von beschnittenen Aufnahmen, deren Hintergrund entfernt und neutral gefüllt wurde.

Die in diesem Abschnitt vorgestellten Module untersuchen die Bilddaten eines

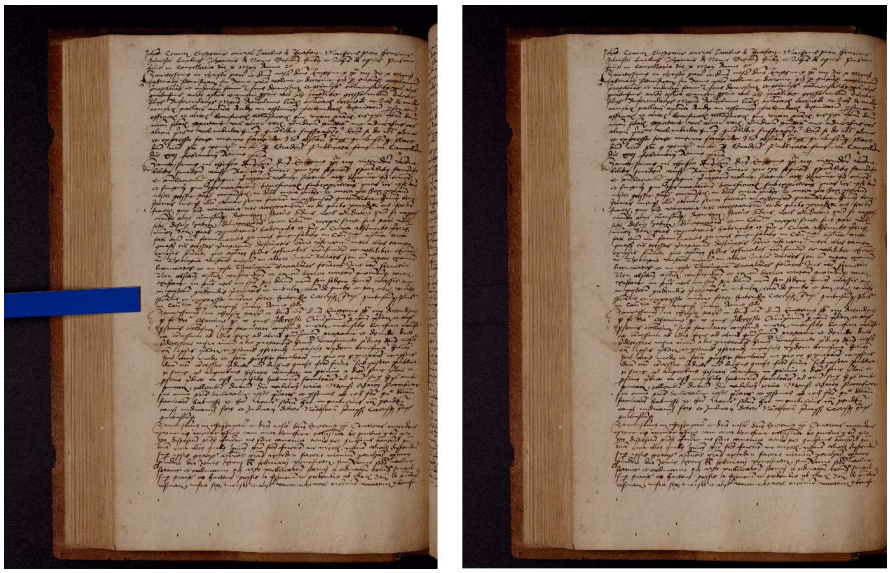


Abbildung 9.7: Analyse-Modul zum Retuschieren von Halterungen.

Digitalisierungsprojekts automatisiert auf Produktionsfehler und sind für das Entwurfsziel, die Durchlaufzeit zu minimieren und die Reproduktionsqualität zu maximieren, essenziell. Damit der Anwender des Workflow-Systems, der für die Übergabe der Bilddaten eines Digitalisierungs-Projekts an die Verarbeitungskette verantwortlich ist, ein zeitnahes Feedback über die Ergebnisse der Analysen erhält, wurden die automatisierten Module als effizient verzahnte Pipeline implementiert. Insbesondere werden dafür die in Abbildung auf Seite 128 beschriebenen *Enterprise Integration Patterns* zur Nachrichten-Steuerung verwendet. Die Analysen laufen damit bereits ab, während der Anwender die Bilddaten noch in das System lädt, was die Zeit des Analyse-Feedback-Zyklus verkürzt und das Risiko, einen Produktionsfehler erst dann zu entdecken, wenn sich die zu digitalisierende Quelle schon wieder am Aufbewahrungsort befindet, verringert.

Automatisierte Metadaten-Extraktion

Wie in Abschnitt 2.2 auf Seite 23 beschrieben, ist es ein erklärtes Ziel der Retro-Digitalisierung, neben einer möglichst originalgetreuen Bilderfassung auch umfassende Metadaten zu einer Quelle bereitzustellen. Um die menschliche Beteiligung im Erfassungs-Prozess möglichst gering zu halten, wurden im Workflow-System unterschiedliche Verfahren aus dem Bereich des Information Retrievals eingesetzt, die eine Metadaten-Extraktion auf der Quelle vornehmen.

Analog zur Analyse der Bilddaten werden diese Module ebenfalls über eine schmale Schnittstelle eingebunden.

Term-Klassifikation durch Auswertung des Kontexts Die in dieser Arbeit beschriebenen Ansätze zur Klassifikation von Termen durch Auswertung der Informationen in ihrem Kontext eigenen sich, um in den syntaktisch anspruchsvollen und von Werk zu Werk wechselnden Domänen angewendet zu werden und den Anwender automatisiert bei der Bereitstellung von beschreibenden Metadaten (siehe Seite 23) zu unterstützen. Das Kontext-Klassifikations-Modul erwartet als Eingabe einen Text, beispielsweise den Volltext oder ein Transkript eines Werks. Wie in den vorherigen Abschnitten beschrieben wird daraufhin versucht, Named Entities wie *Toponyme*, *Personennamen*, *Zeitangaben* oder *Rollen* zu identifizieren und ihre Vorkommen im Text zu kennzeichnen. Dem Anwender des Workflow-Systems wird dadurch die Metadaten-Annotation eines Werks und die Herstellung von Querverweisen zu anderen Werken maßgeblich erleichtert. Am Ende des Kapitels finden sich Beispiele für die Verwendung der so gewonnenen Metadaten.

Semantische Analyse und Annotation historischer Landkarten Historische Landkarten sind eine reichhaltige Informationsquelle für die Forschung: An ihnen lassen sich beispielsweise die sprachliche Entwicklung von Ortsnamen im Laufe der Zeit nachvollziehen oder die Gründung, Verschmelzung aber auch das Verschwinden von ganzen Ortschaften. Weiterhin codieren Landkarten vorrangig geospatiale Abhängigkeiten zwischen den dargestellten Ortsmarkierungen. Um präzise Forschungsarbeit auf Landkarten-Material ermöglichen zu können, sind möglichst

genaue Metadaten notwendig, beispielsweise die räumliche Ausdehnung des dargestellten Kartenausschnitts und die darin enthaltenen Ortschaften. Diese Informationen sind darüber hinaus auch notwendig, um Bezüge zwischen Ortschaften in unterschiedlichen Landkarten herstellen zu können.
Innerhalb des Workflow-Systems lässt sich ein Metadaten-Modul zuschalten, welches dem Anwender ermöglicht, eine Landkarte und die darin enthaltenen Informationen semi-automatisiert zu annotieren. Mithilfe von *Template-Matching* kann das Modul Orte anhand der verwendeten Ortsmarkierungen auf der Landkarte identifizieren. In [Höhn, Schmidt und Schöneberg (2013)] werden das Template-Matching und die Vorzüge des Annotations-Moduls detaillierter beschrieben. Auf diese Weise gefundene historische Ortsbezeichnungen lassen sich innerhalb des Moduls dann auf ihre modernen Entsprechungen querverweisen. Dadurch werden nicht nur abweichende Ortsnamen in verschiedenen Karten über ein zentrales Datum verbunden, es ermöglicht dem Nutzer auch, bei einer Suche nach einem aktuellen Ortsnamen dessen historische Schreibweisen-Varianten und deren Entwicklung nachzuvollziehen.
Wie auch das Kontext-Klassifikations-Modul liefert dieses Modul beschreibende Metadaten, die auf vielfältige Weise nachgenutzt werden können, wie die Beispiele am Ende des Kapitels belegen.

Extraktion des Inhaltsverzeichnisses und der Seitenzahlfolge Ein weiteres Modul zur Metadaten-Extraktion ermöglicht das automatisierte Auffinden von Inhaltsverzeichnissen in digitalisierten Dokumenten. Als Eingabe wird eine textuelle Entsprechung des Werks erwartet (beispielsweise ein existierender Volltext, eine angefertigte Übersetzung oder die Ausgabe von automatisierten Methoden wie Optical Character Recognition (OCR)). Der Algorithmus verwendet domänenspezifische Heuristiken und statistische Evaluationen und versucht, Position und Aufbau des Inhaltsverzeichnisses eines Werks zu analysieren. Weiterführende Informationen zu diesem Verfahren finden sich in [Beretta (2012)].
Die auf diese Weise bereitgestellten Metadaten fallen in die Kategorie der strukturellen Metadaten (vgl. Seite 23) und geben Aufschluss über den Aufbau des Werks. Anhand dieser Informationen kann eine für den Nutzer intuitive Navigati-

on innerhalb des digitalisierten Werks erzeugt werden. Weiterhin erlauben die so gewonnenen Informationen eine Plausibilisierung anderweitiger Informationen, so zum Beispiel die korrekte Abfolge von Seitenzahlen oder eine Dublettenerkennung. In der Virtuellen Bibliothek[9] lassen sich viele Beispiele für navigierbare Digitalisate finden.

Publikation, Langzeitarchivierung und Nachnutzung

Es existiert eine Vielzahl etablierter Metadaten-Portale (z.B. *Open Archives Initiative*[10]), digitaler Sammlungen (z.B. *Europeana*[11]), Langzeit-Archivierungslösungen (z.B. *Rosetta*[12]) und Online-Publishing Plattformen (z.B. *DigiTool*[13]). Wie in Abschnitt 1.1 auf Seite 11 beschrieben ist es im Interesse der Digitalisierungsabteilung, ihre Digitalisate einem breiten Publikum zur Verfügung zu stellen und sie redundant auf mehreren Servern für die Zukunft zu sichern. Aus diesem Grund wurde bei der Entwicklung des Workflow-Systems hoher Wert darauf gelegt, über erweiterbare Module Exporte auf unterschiedlichste Zielplattformen anbieten zu können. Das Workflow-System verwendet für den Metadaten- und Bild-Austausch ein Export-System, welches auf Templates basiert. Ein Template ist hierbei eine mit Platzhaltern markierte Schablone für einen Zielcontainer (beispielsweise TEI: P5, METS/MODS), welche mit den konkreten Metadaten des zu exportierenden Werkes befüllt wird.
Derzeit unterstützt das System bereits Exporte in Langzeit-Archivierungslösungen vom Rechenzentrum der Universität Würzburg, es bietet Metadaten-Exporte nach METS/MODS-Standard und kann neben Publikations-Tools wie dem oben genannten DigiTool-Viewer ein eigenes Publikations-Portal[14] bedienen.

[9] http://vb.uni-wuerzburg.de/
[10] http://www.openarchives.org/ore/
[11] http://europeana.eu
[12] http://www.exlibrisgroup.com/de/category/Rosetta
[13] http://www.exlibrisgroup.com/category/DigiToolOverview
[14] http://vb.uni-wuerzburg.de

Integration

Abbildung 9.8 illustriert, welche Interaktionsmöglichkeiten das Workflow-System bietet: Bereits angesprochen wurde die Möglichkeit, über eine **Client-Anwendung** Zugriff auf das Workflow-System zu erhalten.

Es handelt sich dabei um eine graphische Nutzeroberfläche (eine so genannte *rich client application*), die mittels Java-Web-Start[15] an den Anwender ausgeliefert wird. Voraussetzung dafür ist ein Java Runtime Environment (JRE) der Version 7 oder höher und Anschluss an das Internet. Mit diesem Modell lassen sich bequem externe Anwender, Kunden und Entwickler in den Verarbeitungsprozess integrieren. Über im Workflow-System zu setzende Berechtigungen können die Aktionen der jeweiligen Nutzer von den Administratoren angepasst werden. Lädt ein Nutzer mit Hilfe der Client-Anwendung Bild- oder Metadaten zu einem Digitalisierungsprojekt auf den Server, werden dort die jeweiligen Verarbeitungsketten angestoßen.

Eine zweite Möglichkeit zur Kommunikation stellt die **REST-Schnittstelle** (*representational state transfer* wie beschrieben in [Fielding (2000)]) des Workflow-Systems dar: Mit Hilfe von vier HTTP-Methoden *GET, POST, UPDATE, DELETE* lassen sich über eine definierte URL die Datenbanken des Workflow-Systems abfragen, Projekte anlegen, löschen und auch Metadaten pflegen. Als Reaktion auf eine per REST eingehende Anfrage kann auch die Verarbeitungskette für Bild- und Metadaten im Workflow-System angestoßen werden. Die REST-Schnittstelle bietet einen einfachen Ansatzpunkt für externe Mitarbeiter und Entwickler, eigene Daten in das System einzuspeisen oder aktuelle Statistiken zu generieren. Abbildung 9.14 zeigt beispielhaft die Ausgabe einer Webanwendung, die eine Statistik zur Nutzung des Workflow-Systems aus den Daten der REST-Schnittstelle generiert.

Die dritte Möglichkeit zur Interaktion mit dem Workflow-System ist die direkte Kommunikation über den zentralen **Nachrichten-Aktor** der Server-Komponente.

[15] http://www.oracle.com/technetwork/java/javase/javawebstart/index.html

Die Vorzüge der dabei verwendeten Techniken werden in Kapitel 6.2 auf Seite 64 detailliert beschrieben. Entwickler haben durch diesen Kommunikations-Endpunkt die Möglichkeit, eigene Anwendungen zur Interaktion mit dem Workflow-System zu schreiben oder Webanwendungen, die über das klassische *request-response Modell* des zuvor beschriebenen REST-Ansatzes hinausgehen und eine bidirektionale, asynchrone Kommunikation mit der Server-Komponente des Workflows anstreben.

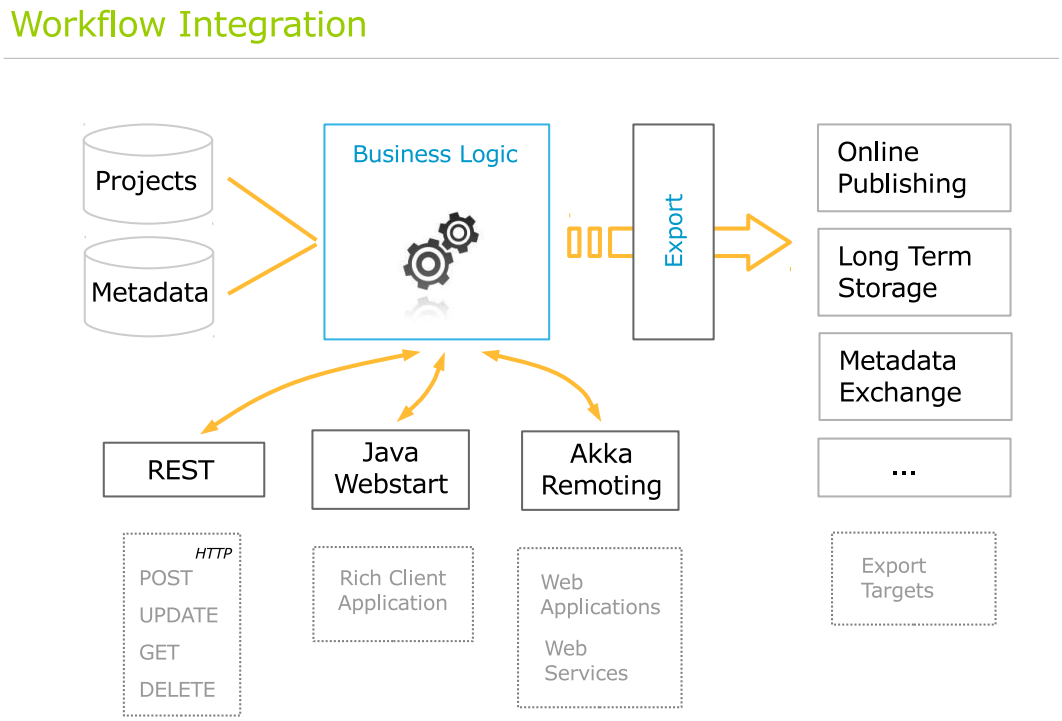


Abbildung 9.8: Workflow Integration Konzept.

Mit den in diesem Abschnitt vorgestellten automatisierten Modulen werden die Entwurfsziele erreicht, die Produktionsfehlerrate zu minimieren, die Durchsatzrate zu maximieren sowie die erstellten Digitalisate einerseits in Langzeit-Archivierungs-Lösungen für die Nachwelt zu erhalten, sie andererseits aber auch einem breiten Publikum auf Publikations-Plattformen zugänglich zu machen. Die dafür notwendigen Analyse- und Metadaten-Extraktions-Verfahren erzeugen jedoch eine hohe Rechenlast. Bevor die Leistungsfähigkeit des Systems abschließend anhand von

Beispielen belegt werden kann, soll nachfolgend zunächst beschrieben werden, wie die beschriebenen Verfahren effizient umgesetzt werden konnten. Dazu wird auf den Begriff der Skalierbarkeit eingegangen, welcher für die Anpassbarkeit eines Systems an sich verändernde Leistungsparameter steht.

Skalierbarkeit

Der Begriff Skalierbarkeit bezieht sich auf die Fähigkeit eines Systems, bei gesteigerten Anforderungen leicht erweitert werden zu können und weitere Ressourcen wie zusätzliche Rechenleistung oder mehr Arbeitsspeicher verwenden zu können. Dabei wird unterschieden zwischen *'scaling up'*, der vertikalen Skalierung, bei der die Leistungssteigerung durch Verwendung größerer oder schnellerer Ressourcen innerhalb des Systems erzielt wird (beispielsweise Austausch des Prozessors gegen ein Modell mit höherer Taktrate, Verwendung von mehr Arbeitsspeicher) und *'scaling out'*, der horizontalen Skalierung, bei welcher zusätzliche Recheineinheiten die Aufgaben des Systems im Verbund bearbeiten. Zu der Wichtigkeit von Skalierbarkeit finden sich in [Michael u. a. (2007)] detailliertere Ausführungen.

In dem vorgestellten Workflow-System muss jedes Digitalisierungs-Projekt automatisierte Qualitäts-Analyse- und Informations-Extraktions-Zyklen durchlaufen (wie in Abbildung 9.2 zu sehen). Da die Bilddaten, die in der Digitalisierung zum Einsatz kommen, üblicherweise in Form eines Rasters gehalten werden [*Tiff 6.0 Image Specification*], können die anfallenden Arbeiten effizient parallelisiert werden, indem man die Bilddaten in Kacheln aufteilt und die Analyse auf den Kacheln parallel ausführt. Dadurch wird Skalierbarkeit zu einem zentralen Element in der Implementierung des Workflow-Systems.

Das Workflow-System setzt Akkas Aktor-Modell ein, um die Bildverarbeitungs- und Metadaten-Extraktions-Aufgaben effizient zu parallelisieren und eine hohe Skalierbarkeit (sowohl horizontal als auch vertikal) der Anwendung zu erreichen.

Die dabei erzeugte Aktor-Hierarchie kann mehrere Rechner überspannen und erzeugt so ein verteiltes Szenario, welches hohe Rechner-Lasten bei Bedarf ausgleichen kann.

Im Digitalisierungs-Workflow sind die Zwischenergebnisse der einzelnen Arbeitsschritte durch die verwendeten Interfaces genau definiert. Mit Hilfe der monadischen Futures, wie sie in Abschnitt 6.5 erläutert werden, kann der komplette, abstrakte und asynchron ablaufende Klassifikations-Workflow abgebildet werden, ohne dass die erst zur Laufzeit ausgewählten Arbeitsschritte bekannt sein müssen. Mit Hilfe dieses Verfahrens lässt sich das Entwurfsziel, je nach Digitalisat und dessen Verwendungszweck eine dynamische Verarbeitungskette zu bilden, ideal umsetzen. Über die abstrakte Formulierung der Abfolge von Arbeitsschritten ist nicht nur für eine leichte Anpassbarkeit des gesamten Digitalisierungs-Workflows an unterschiedliche Domänen gewährleistet, sondern auch die zukünftige Erweiterbarkeit um zusätzliche Module, die auf diesem Wege leicht in die Verarbeitungskette eingegliedert werden können. Darüber hinaus lassen sich damit unabhängige Arbeitsschritte effizient parallel und in einem verteilten Szenario berechnen.

Das Master-Worker-Muster

In Abschnitt 9.4 wurde das *Master-Worker-Muster* bereits vorgestellt, weil es eine auch unbekannten Fehlern gegenüber robuste Architektur des Workflow-Systems ermöglicht. In [Tiona (2013)] werden die Vor- und Nachteile unterschiedliche Umsetzungen dieses Musters beschrieben.

Dieses Muster hat allerdings noch weitere Vorzüge: Es ist dank der referenziellen Transparenz (wie in Abschnitt 6.3 auf Seite 68 beschrieben) von Akkas Aktoren topologie-agnostisch und erlaubt es beliebigen, im Netzwerk befindlichen Arbeiter-Aktoren, Arbeitspakete von einem zentralen Master-Aktor zu erfragen und diese zu bearbeiten. Die Verwendung dieses Musters hat die Ausführungszeit von Bildbearbeitungs- und Metadaten-Extraktions-Aufgaben deutlich reduziert, wie Abbildung 9.9 belegt: Das Beispiel verdeutlicht die Leistungssteigerung, die durch den Einsatz des Master-Worker-Musters mit Einbindung verteilter Rechner

gegenüber einer strikt auf einen Rechner beschränkten Variante am Beispiel einer Bildskalierungs-Aufgabe erzielt werden konnte. In Abbildung 9.10 wird die Hierarchie der am Muster beteiligten Aktoren dargestellt.

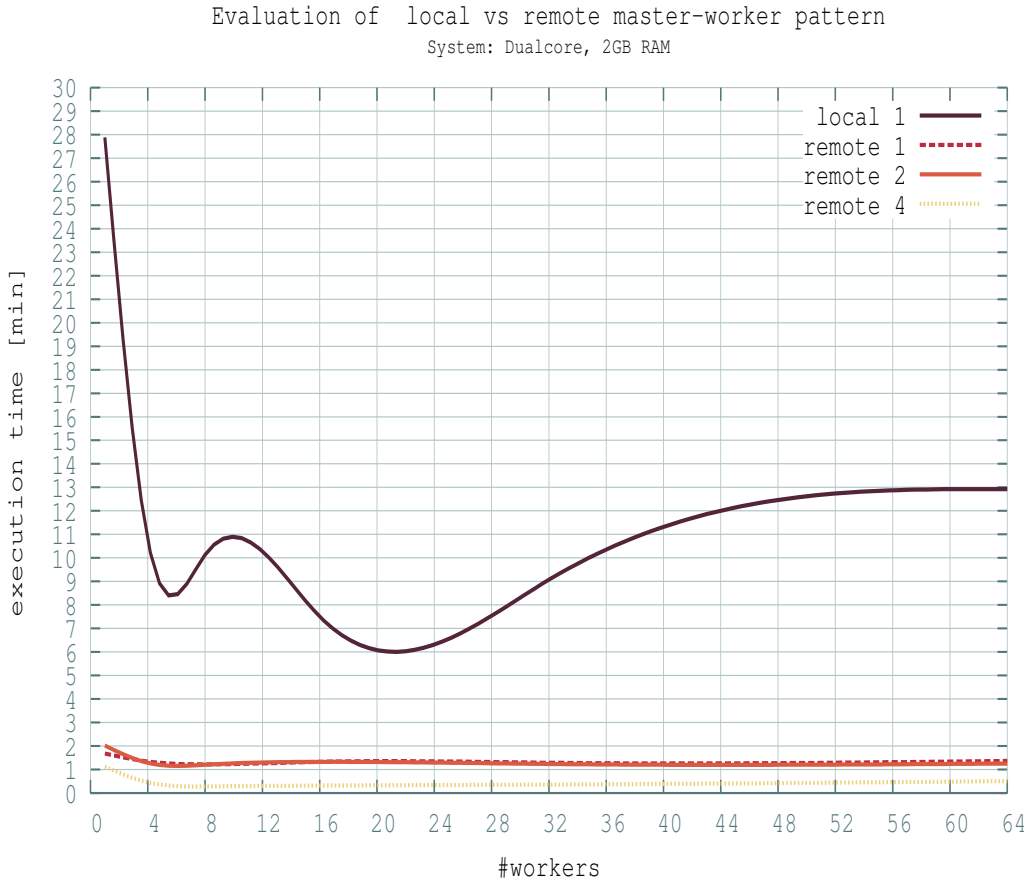


Abbildung 9.9: Vergleich von lokalem und verteiltem Rechenszenario. Quelle der Messwerte: [Tiona (2013)]

Durch diesen Mechanismus kann die geforderte Skalierbarkeit des Systems leicht umgesetzt werden: Das Workflow-System besteht aus einer zentralen Server-Applikation, die den Master-Aktor sowie einige Arbeiter-Aktoren zur Verfügung stellt,

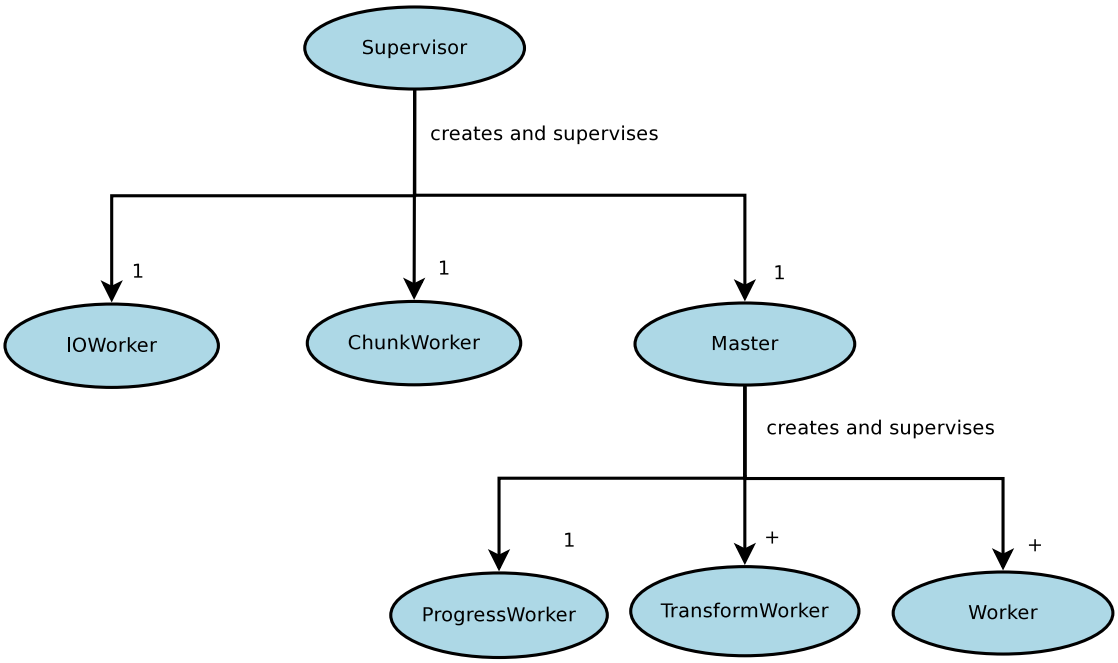


Abbildung 9.10: Variante des master-worker-Musters. Schematische Darstellung der beteiligten Aktoren. Quelle: [Tiona (2013)]

und einer Client-Applikation, die den Nutzern zur Verfügung gestellt wird. Beim Start der Client-Anwendung können (je nach Leistungsfähigkeit der ausführenden Maschine) Arbeiter-Aktoren auf dem Rechner des Anwenders erzeugt werden, welche sich dann als zusätzliche Arbeiter beim zentralen Master-Aktor registrieren können. Diese Herangehensweise erlaubt es, die am Digitalisierungs-Prozess beteiligten Rechner mit in die Bewältigung der anfallenden Arbeitspakete einzubeziehen.

Gegenüber der in [Tiona (2013)] evaluierten Version bietet die im Workflow-System eingesetzte Version zusätzlich noch die Möglichkeit, die maximale Dauer einer Aufgabe zu definieren. Sollte diese überschritten werden, verteilt der Master-Aktor die bis dahin nicht erfüllte Aufgabe anderweitig.

9.5 Use-Case: OCR Verarbeitung im Workflow-System

Das Workflow-System sieht die Metadaten-Verarbeitung und -Nachnutzung explizit in den Verarbeitungsschritten eines Digitalisierungs-Projekts vor. Nachfolgend soll aufgezeigt werden, welche Schritte notwendig sind, um zusätzliche Module zur Verarbeitungskette hinzuzufügen.

9.5.1 Beschreibung der Metadaten-Verarbeitungskette im Workflow-System

Das Workflow-System setzt auf eine auf einfachen Nachrichten basierende Kommunikation zwischen allen beteiligten Komponenten (*message driven design*). Die einzelnen Komponenten können dadurch entkoppelt voneinander mit unterschiedlichsten Technologien entwickelt werden, da sie als gemeinsame Schnittstelle lediglich Textnachrichten besitzen.

Für das Versenden von Nachrichten existieren bereits vielfältige erprobte Lösungen. Analog zu Entwurfsmustern im klassischen Software-Entwurf finden sich in (Hohpe und Woolf 2003) viele Entwurfsmuster - so genannte Enterprise Integration Patterns (EIP)[16] - für Szenarien rund um das Versenden von Nachrichten. Innerhalb des Workflow-Systems wurde zunächst eine *message oriented middleware* (MOM) eingesetzt, die nach dem Java Message Service[17] Standard arbeitet. Es handelt sich dabei um eine externe Komponente, die sich um das Versenden der von den Komponenten erzeugten Textnachrichten kümmert. Die Nachrichten werden nicht nur zwischen den Komponenten verschickt, auch zur Kommunikation zwischen Server- und Client-Komponente kommen sie zum Einsatz. Aktuell werden die Nachrichten unter Verwendung des Aktoren-Modells versendet. Das

[16] http://www.eaipatterns.com/eaipatterns.html
[17] https://java.net/projects/jms-spec/pages/Home/

Modell und die vielfältigen Vorzüge, die es bietet, werden in Abschnitt 6.2 genau beschrieben.

Von der Nachrichtenebene wird dann mit Hilfe von Apache Camel[18] abstrahiert, einem Framework, welches die meisten der EIPs implementiert, was ein komplexes und dynamisches Versenden von Nachrichten zwischen den Programmteilen erlaubt. Für den Entwickler ergibt sich daraus der Vorteil, dass die technischen Details der verwendeten MOM aus der Geschäftslogik seines Moduls wegabstrahiert werden und er sich lediglich um das Erzeugen und Verarbeiten von Textnachrichten kümmern muss. Die Entwickler des Workflow-Systems erhalten andererseits vielfältige Möglichkeiten durch die Verwendung der EIPs wie beispielsweise *load balancing* (Ansteuerung verschiedener Komponenten bei unterschiedlicher Systemauslastung), *content-based routing* (Ansteuerung von Komponenten basierend auf Nachrichteninhalt) oder *wire tapping* (beispielsweise automatisierte Buchhaltungsfunktionen bei Verwendung bestimmter Komponenten).

Abbildung 9.4 veranschaulicht, wie mit Hilfe des EIP: Pipes and Filters[19] zur Laufzeit genau die Komponenten zu einer dynamischen Verarbeitungskette zusammengesetzt werden können, die für den Typ des vorliegenden Digitalisats und dessen Verwendungszweck passend sind. Die angesteuerten Komponenten wissen dabei nicht, dass sie Teil einer Verarbeitungskette sind. Die Liste der zum Einsatz kommenden Module ist beliebig erweiterbar. Im weiteren Verlauf wird beschrieben, welche Schritte notwendig sind, um eigene Module zur Metadatenverarbeitung zur Produktionskette hinzuzufügen.

9.5.2 Implementierung

Im Wesentlichen existieren drei zentrale Schnittstellen, über die das Workflow-System erweitert werden kann: `AnalysisModule` (siehe Code-Beispiel 9.2) ist die Schnittstelle, welche von Modulen zur automatisierten Kontrolle von Bilddaten implementiert werden muss, `IEModule` (siehe Code-Beispiel 9.3) ist die Schnittstelle, die von Modulen zur automatisierten Extraktion von Metadaten implementiert

[18] http://camel.apache.org/
[19] http://www.eaipatterns.com/PipesAndFilters.html

werden muss und `ExportModule` ist die abstrakte Oberklasse aller Export-Module, also aller Module, die ein digitalisiertes Projekt in ein anderes Format exportieren, es auf einer Webseite publizieren oder auf andere Weise bereitstellen wollen.

```
0  public abstract class AnalysisModule implements Runnable {
1    protected BufferedImage img;
2    protected File file;
3    protected ActorRef resultMonitor;
4    protected String moduleName = "NOTSET";
5
6    /**
7     * this method will execute the actual image analysis
8     *
9     * @param pic - the image to analyze
10    * @param filename - the name of the analyzed file
11    */
12   protected abstract String analyze(BufferedImage pic, File file);
13
14   /**
15    * short description of the module, eg. 'ThumbnailCreation'
16    *
17    * IMPORTANT: This String works as module identifier and should not contain
18    * whitespaces or non-word-chars (as regexp-matched by \W)
19    *
20    * valid examples: ThumbnailCreation, PageNumberAnalysis,
21    * MarginErrorDetection ...
22    *
23    */
24   public String getModuleName() {
25     return moduleName;
26   }
27
28   @Override
29   public void run() {
30     resultMonitor.tell(new AnalysisResult(this.file, this.getModuleName(),
31         analyze(img, file)));
32   }
33 }
```

Listing 9.2: Interface 'AnalysisModule' für Module zur automatisierten Qualitätskontrolle.

Alle Module, die eine der drei genannten Schnittstellen implementieren, müssen lediglich zur Laufzeit im Klassenpfad bereit liegen und eine *provider-configuration* Datei im Verzeichnis `META-INF/services` bereitstellen (siehe *Oracle ServiceLoader<S> API Dokumentation*). Sie werden dann über den *Service Provider* Mechanismus vom Workflow-System gefunden, ohne dass weitere Anpassung von Seiten der Entwickler notwendig wären. Nach (*Oracle ServiceLoader<S> API Dokumentation*) stellen die beschriebenen Schnittstellen und abstrakten Klassen einen *Service* dar, die Klassen, die diese implementieren oder erweitern, werden *Service Provider* genannt. Über einen `ServiceLoader<S>` für einen Service vom Typ `S`

können dann zur Laufzeit Service Provider für den Service `S` gefunden werden.

```
0  package de.hensb.analysismodule;
1
2  import java.io.File;
3
4  /**
5   * in difference to the {@link AnalysisModule} classes implementing this
6   * interface are supposed to perform information retrieval / extraction tasks.
7   *
8   */
9  public interface IEModule {
10
11   public abstract String analyze(final File root)
12       throws Exception;
13
14   public abstract String getModuleName();
15 }
```

Listing 9.3: Interface 'IEModule' für Module zur automatisierten Metadaten-Extraktion.

9.5.3 Erweiterungen des Workflow-Systems

In den folgenden Abschnitten soll beschrieben werden, wie Entwickler eigene Module in die Verarbeitungskette einbringen können. Dieses wird anhand von vier beispielhaften Use-Cases illustriert.

Alignment von Pixel-Bild und OCR-Text

Die genaue Zuordnung von Bild- und Volltextdaten ist gerade auf historischen, fehlerträchtigen Texten ein schwieriges Problem. Dennoch ist sie nicht zuletzt deswegen erstrebenswert, weil sich für den Betrachter mit Hilfe einer synoptischen Gegenüberstellung (wie in Abbildung 9.11 veranschaulicht) ein deutlicherer Bezug zwischen Bild und Text herstellen lässt.

Um das Workflow-System um ein Modul für das Alignment von Bild- und Volltextdaten zu erweitern, muss eine neue Implementierung der `IEModule`-Schnittstelle bereitgestellt werden. Durch den zuvor beschriebenen Service Provider Mechanismus muss diese Implementierung lediglich (beispielsweise in Form einer Jar-Datei)

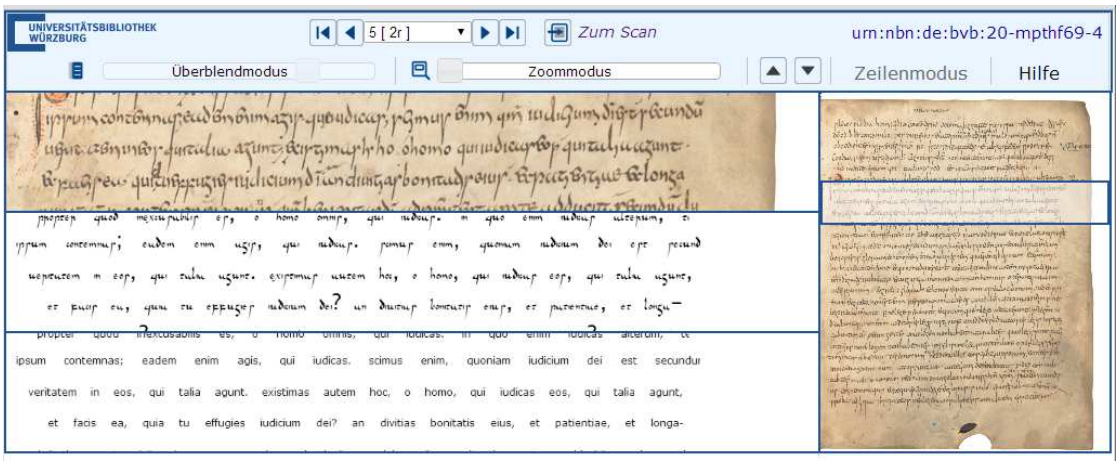


Abbildung 9.11: M.p.th.f.69 Bild- und Volltext-Überlagerung. Quelle: Virtuelle Bibliothek Würzburg

zur Laufzeit im Klassenpfad verfügbar gemacht werden und wird dann automatisch gefunden und eingebunden. Alle `IEModule`-Implementierungen werden jeweils dann aktiv, wenn Metadaten wie Volltexte über das Workflow-System zu einem Digitalisierungsprojekt hinzugefügt werden.

Die Geschäftslogik des neuen Moduls sollte zunächst versuchen, Zeilen im Bild zu erkennen und anschließend ein Alignment mit den Zeilen des Volltextes erzeugen. Zur Zeilenerkennung im Bild sollten Tools wie *Tesseract OCR* (*Tesseract OCR*) oder *ocropus* (*ocropus*) verwendet werden, da diese neben der reinen Zeichensequenz-Erkennung auch Layout-Informationen zu den untersuchten Regionen im Bild liefern (beispielsweise Bounding Boxes).

Das Modul sollte dann aus den im Bild erkannten Zeilen und den Zeilen des Volltexts ein Alignment generieren. Das kann einerseits ohne Nutzerbeteiligung passieren und automatisiert im Hintergrund ablaufen. Möglicherweise ist aber gerade bei schwierigem Bildmaterial eine Nutzerbeteiligung am Alignment-Prozess wünschenswert. Für diesen Fall sollte das Workflow-System um eine spezialisierte Viewer-Komponente erweitert werden, die nur dann verwendbar ist, wenn Bild- und Volltext-Daten vorhanden sind und die Analyse durch das zuvor bereitgestell-

te Alignment-Modul abgeschlossen ist.

Einbindung eines Editors für OCR-Texte

Das Workflow-System behandelt Metadaten bereits jetzt als expliziten Baustein in der Verarbeitungskette eines Digitalisats. Über die Client-Komponente oder über die REST-Schnittstelle können werksbegleitende Metadaten eingepflegt werden.

Häufig ist es wünschenswert, die hinzugefügten Metadaten vom Nutzer editieren zu lassen - um beispielsweise OCR-Fehler zu korrigieren oder annotierte Texte zu überarbeiten. Dazu bieten sich grundsätzlich zwei Wege an:

Die eingangs beschriebene **REST-Schnittstelle** bietet die Möglichkeit, die begleitenden Metadaten eines Digitalisats unter Verwendung von vier HTTP-Befehlen (*GET, POST, DELETE, UPDATE*) abzufragen, hinzuzufügen, zu löschen oder zu verändern. Denkbar wäre also ein externer Web-Service oder eine Anwendung, die über die REST-Schnittstelle zunächst die zu editierenden Metadaten eines Werks abfragt und sie dem Nutzer in einer geeigneten Editionsoberfläche präsentiert. Nach Änderung der Metadaten können diese über die REST-Schnittstelle wieder ins System eingespeist werden. Dazu muss die REST-API unter Umständen für spezifische Aufgaben angepasst werden. Code-Beispiel 9.4 zeigt einen Ausschnitt der in Scala unter Verwendung von *Spray*[20] geschriebenen REST-API für die Abfrage von Digitalisierungsprojekten.

[20] http://spray.io

```
/**
   * all project routes
   */
  def projectRoutes(api: WuesyphusRESTAPI) =
    get {
      pathPrefix("projects") {
        pathEnd {
          // FIND BY STATUS
          parameters('active.as[Boolean])(active =>
            findByStatus(api, active)) ~
            // FIND ALL
            findAll(api)
        } ~
          // FIND BY YEAR ...
          pathPrefix("\\d{4}".r) {
            year =>
              // ... AND MONTH ...
              pathPrefix("\\d{2}".r) {
                month => pathEnd(findByDate(api, year, month))
              }
          } ~
          // FIND BY ORDER
          pathPrefix("\\d+".r) { order =>
            pathEnd(findByOrder(api, order)) ~
              // FIND PROJECT DIR
              path("dir")(findProjectDirByOrder(api, order)) ~
              // FIND MANIFESTS BY ORDER
              pathPrefix("manifests") {
                pathEnd((findManifestsByOrder(api, order))) ~
                  pathPrefix(".+".r) { dir =>
                    // FIND MANIFEST FILES BY ORDER
                    pathEnd(findManifestFiles(api, order, dir)) ~
                      path(".+".r)(file
                        => findManifestFile(api, order, dir, file))
                  }
              }
          } ~
          // FIND BY TYPE
          path("type" / "\\w+".r)(mtype => findByType(api, mtype))
      }
    }
```

Listing 9.4: REST-API für Abfragen von Digitalisierungs-Projekten.

Weiterhin könnte auch die Client-Komponente des Workflow-Systems um eine optionale grafische Komponente erweitert werden, die bei Vorhandensein von Volltextdaten deren Änderung direkt im Workflow-System erlaubt.

Integration von Qualitätsmanagement-Funktionen auf OCR-Texten

Für Bilddaten ist - wie in Kapitel 9.4 ausführlich beschrieben - eine Verarbeitungskette von automatisierten Qualitätsmanagement-Modulen implementiert, die dann aktiv wird, wenn Nutzer über das Workflow-System Bilddaten zu einem Digitalisierungsprojekt hinzufügen. Ähnlich verhält es sich mit Metadaten: Kapitel

9.4 beschreibt die im Workflow-System vorhandenen Module zur automatisierten Extraktion von Metadaten. Auch in diesem Fall wird die Verarbeitungskette angestoßen, wenn Nutzer über das Workflow-System Metadaten zu einem Digitalisierungsprojekt hinzufügen.

Weitere Module zur Analyse von Metadaten können nun wie in Abschnitt 9.5.2 beschrieben hinzugefügt werden.

Da auch über die REST-Schnittstelle Metadaten hinzugefügt werden können, kann auch diese um Analyse-Funktionen erweitert werden, die nach Veränderung der Daten ausgeführt werden. Dazu sollte ein Ansatz analog zu dem Workflow-System verwendeten Service Provider Mechanismus verwendet werden.

Integration von Metadaten auf OCR-Texten

Zumeist handelt es sich bei mit Hilfe von OCR gewonnenen Texten um eine Arbeitsgrundlage, auf deren Basis menschliche Experten und automatisierte Verfahren ihre Arbeit beginnen können. In dieser Arbeit wurden Verfahren zur Named Entity Recognition auf syntaktisch schwierigen historischen Texten mit dem Ziel der Identifikation von Toponymen vorgestellt.

Da OCR-Texte schon in die Kategorie von beschreibenden Metadaten (wie definiert auf Seite 23) zu einem Dokument fallen, sind Toponymfunde innerhalb von Volltexten genau genommen Meta-Metadaten, die ihrerseits in einem geeigneten Format gespeichert werden müssen. Dazu bietet sich beispielsweise das XML-Format *TEI: P5* der *Text Encoding Initiative*[21] an, welches eine äußerst detaillierte Annotation von Texten erlaubt. Das Format wird in (*TEI: P5 Guidelines*) genau beschrieben. Weitere bekannte Metadaten-Formate sind Dublin Core[22] und METS[23]/MODS[24].

[21] http://www.tei-c.org/index.xml
[22] http://dublincore.org/
[23] http://www.loc.gov/standards/mets/
[24] http://www.loc.gov/standards/mods/

Das Workflow-System kann nun wieder an den zwei bereits beschriebenen Stellen erweitert werden, um Volltexte im Format TEI: P5 zu editieren und zu speichern: Einerseits kann ein externer (Web-) Service die Volltext-Daten eines Digitalisats über die REST-Schnittstelle abholen, sie lokal modifizieren und danach über die REST-Schnittstelle wieder einspeisen, andererseits kann das Workflow-System selbst um eine grafische Komponente erweitert werden, die das Editieren von bestehendem Volltext und das Speichern im TEI: P5-Format ermöglicht. Zur Validierung der Daten existieren Schemata (beschrieben in (*TEI: P5 Guidelines*)). Die Implementierung der notwendigen grafischen Komponente für die Textbearbeitung und Annotation könnte mit Hilfe des XML-Editors *<oxygen/>*[25] erfolgen, da dieser einerseits bereits TEI: P5 unterstützt, andererseits gibt es für ihn Plugins, die eine Entwicklung aus einer Java-Umgebung heraus vereinfachen.

[25] http://www.oxygenxml.com/

9.6 Evaluation

Der nachfolgende Abschnitt belegt den Effekt des Workflow-Systems anhand von Nutzungs-Statistiken, qualitativen Analysen und gibt Beispiele für eine Nachnutzung der im Workflow gesammelten Metadaten in der Präsentations-Schicht.

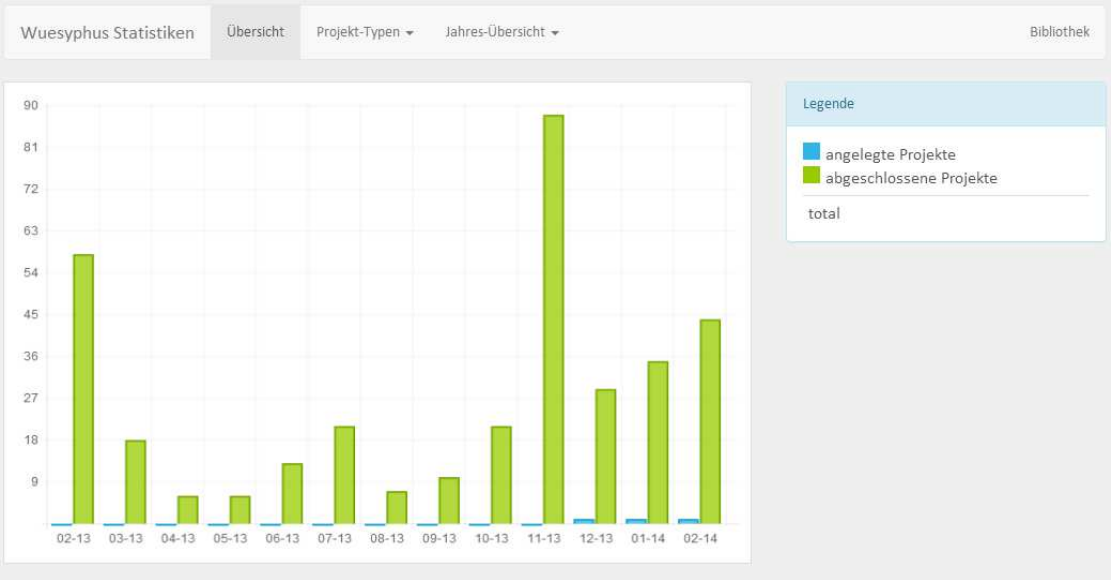


Abbildung 9.12: Ausgabe des Statistik-Moduls zum Workflow-System. Ansicht zeigt angelegte und abgeschlossene Digitalisierungsaufträge des letzten Jahres.

Workflow-System

Mit dem Workflow-System wurden insgesamt etwa 600 Digitalisierungsprojekte erfolgreich abgeschlossen, davon in einem Zeitraum von 12 Monaten ca. 400 (Stand Februar '14, siehe Abbildung 9.12). Daraus ergibt sich eine Produktionsrate von im Schnitt mehr als einer abgeschlossenen Digitalisierung pro Tag, heruntergebrochen auf Arbeitstage werden sogar zwei Digitalisate pro Tag fertiggestellt. Ein typisches Digitalisierungsprojekt hat im Schnitt 300 Aufnahmen, wobei jede Aufnahme mit durchschnittlich 200 MiB Datenvolumen zu Buche schlägt.

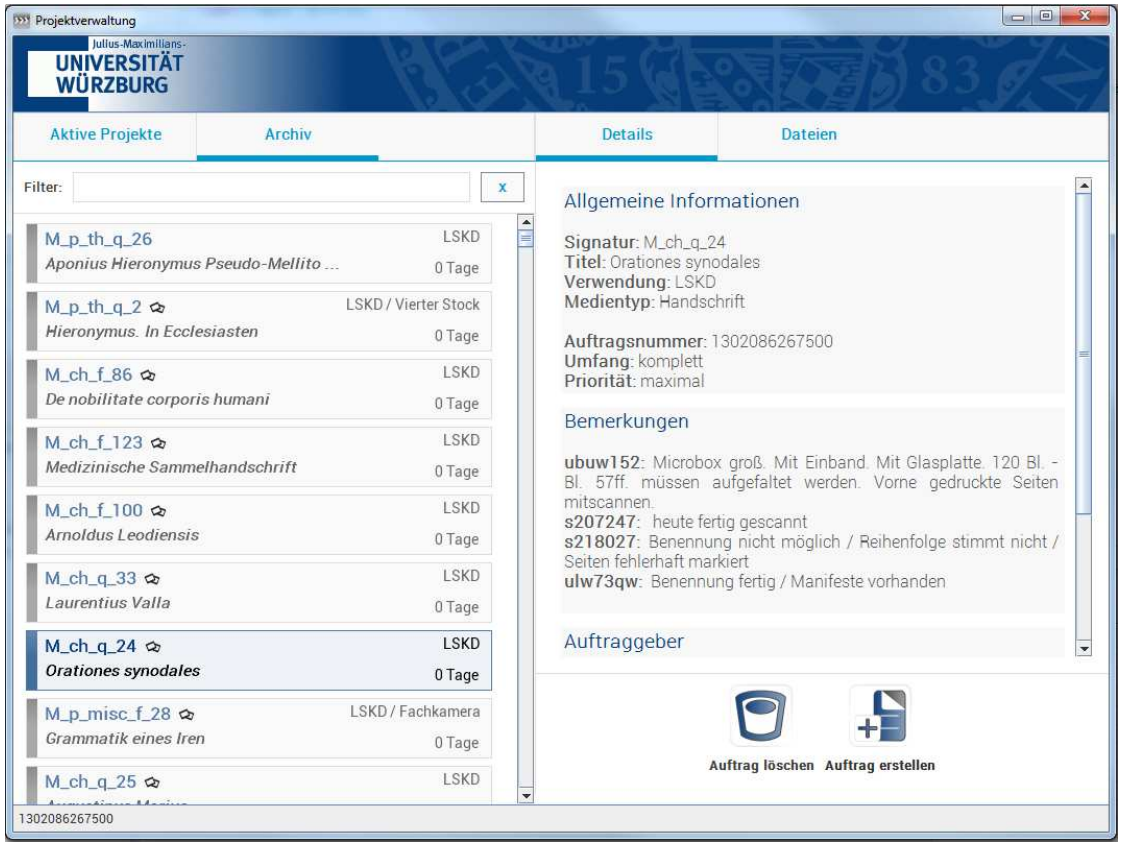


Abbildung 9.13: Übersicht über aktuelle und abgeschlossene Digitalisierungsprojekte im Workflow-Client.

Abbildung 9.13 zeigt die Projekt-Ansicht in der Client-Komponente des Workflow-Systems. Zu sehen sind die derzeit aktiven sowie die erfolgreich abgeschlossenen und archivierten Digitalisierungsprojekte, welche dem Digitalisierungszentrum weiterhin für den automatisierten Export (beispielsweise auf Online-Publikations-Plattformen), die Auslieferung an Kunden oder nachträgliche Pflege- und Anreicherungs-Arbeiten zur Verfügung stehen.

Abbildung 9.14 zeigt die Ausgabe eines Statistik-Moduls, welches Aussagen über die Verwendung des Workflow-Systems treffen kann. Zu sehen ist beispielhaft die Zusammensetzung der bisher abgeschlossenen Digitalisierungsprojekte nach Dokumenttyp.

Die Mitarbeiter des Digitalisierungszentrums haben im Vergleich zur Produkti-

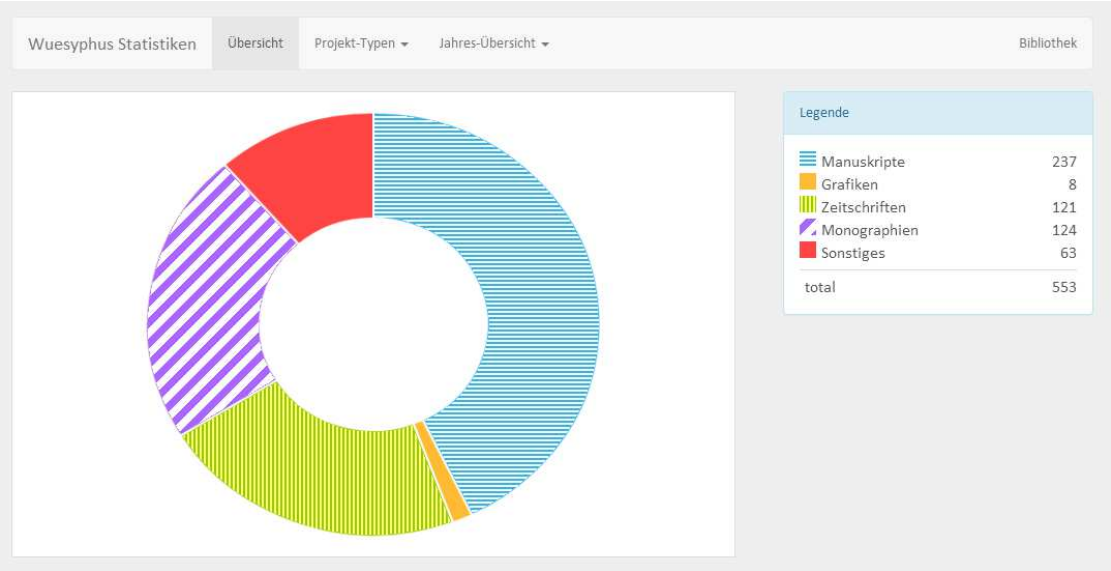


Abbildung 9.14: Ausgabe des Statistik-Moduls zum Workflow-System. Ansicht zeigt Typen aller abgeschlossenen Digitalisierungsaufträge.

onskette ohne Workflow-System eine **Verringerung der Durchlaufzeit um 50 %** [Schmidt (2014)], [Dittrich und Weinmann (2014)] festgestellt. Besonders ausschlaggebend sei nach Angaben der Mitarbeiter die Reduktion der Produktionsfehler durch die automatisierten Module für Qualitätsmanagement und Metadaten-Extraktion (wie in Abschnitt 9.4 besprochen), welche andernfalls in zeitaufwändigen Prozessen nachträglich ausgebessert hätten werden müssen. Wie Abbildung 9.15 zeigt, werden die Ergebnisse der automatisierten Analyse-Module dem Nutzer in der Client-Komponente des Workflow-Systems präsentiert und können helfen, etwaige Fehlerfälle leichter zu identifizieren. Im Beispiel ist zu erkennen, dass das ausgewählte Bild von einer Heuristik zur Dateigrößen-Überprüfung als verdächtig markiert und dem Nutzer zur Inspektion vorgelegt wurde.

Abbildung 9.16 zeigt eine Ansicht aus der Client-Komponente des Workflow-Systems, die es dem Nutzer ermöglicht, einzelne Aufnahmen mit strukturellen Metadaten zu annotieren, beispielsweise der Seitenzahl oder einer Funktion (wie Titelblatt, Einband, etc.). Dabei erhält der Nutzer Unterstützung von den automatisierten Modulen zur Metadaten-Extraktion, wie sie in Abschnitt 9.4 beschrieben werden. Die auf diese Weise gesammelten Metadaten können auf unterschiedliche Weise

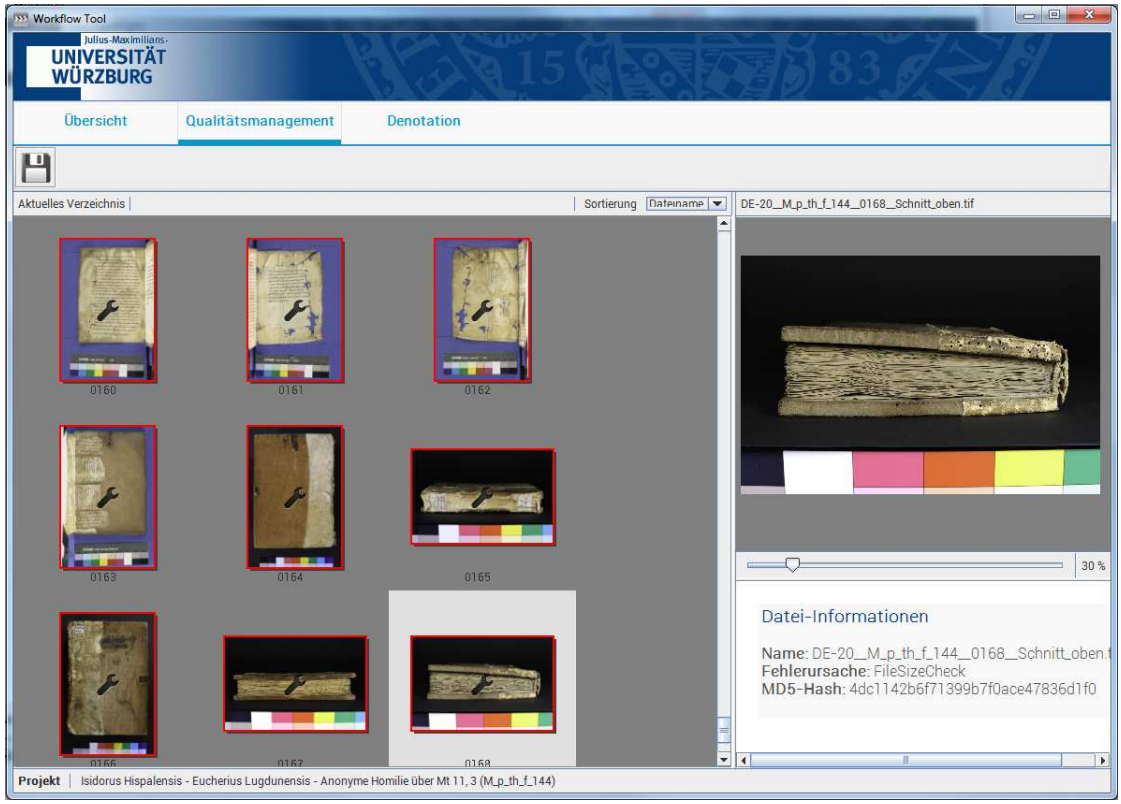


Abbildung 9.15: Qualitätsmanagement in Workflow-Client.

nachgenutzt werden, wie die nachfolgenden Abschnitte belegen.

Abbildung 9.17 zeigt eine Komponente aus dem Workflow-System, die es dem Anwender ermöglicht, ein abgeschlossenes Digitalisierungs-Projekt auf der Präsentations-Plattform *Virtuelle Bibliothek*[26] zu veröffentlichen. Die dafür notwendigen Datenstrukturen, die Erstellung von konvertierten und skalierten Bilddaten, Einbettung von Wasserzeichen zum Schutz vor Missbrauch sowie Kopier-Vorgänge übernimmt das Modul automatisiert.

Nachfolgend soll nun gezeigt werden, wie die hochqualitativen Digitalisate und die im Workflow-System gesammelten Metadaten unterschiedliche Präsentationsmodi erlauben, die sich deutlich von konkurrierenden Systemen abheben. Insbesondere soll die Abgrenzung zu auf Masse und Geschwindigkeit ausgelegten Initiativen wie

[26] http://vb.uni-wuerzburg.de

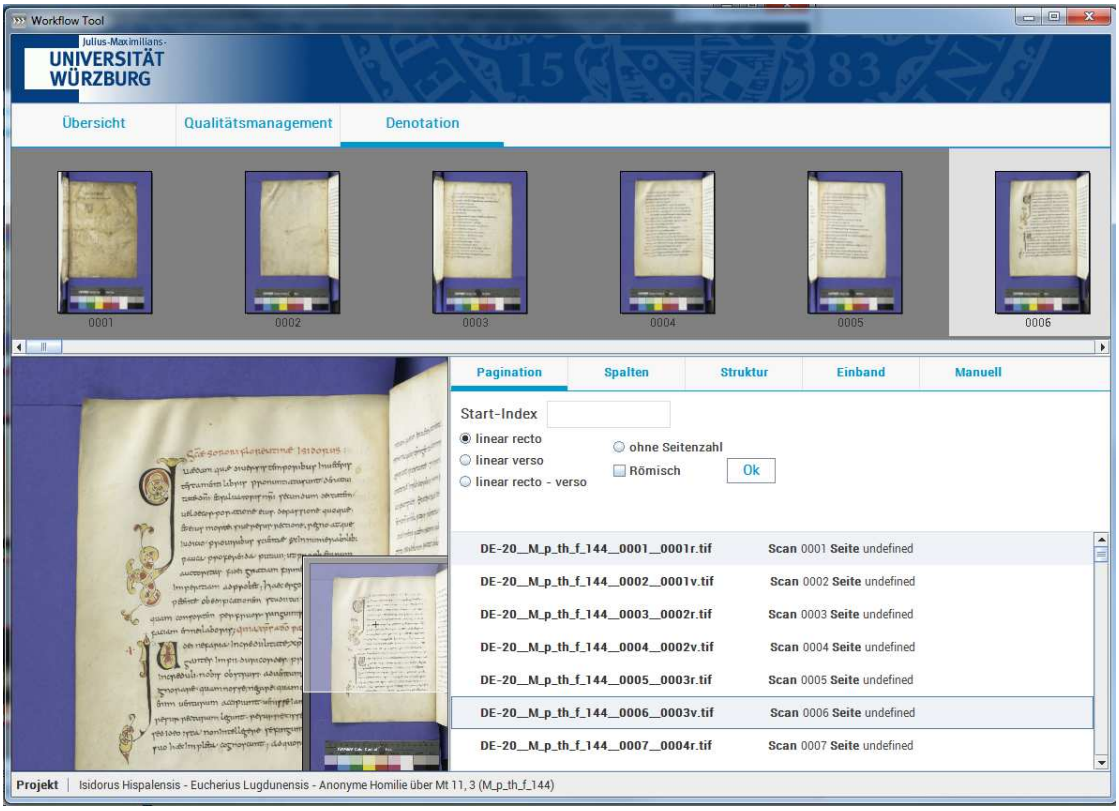


Abbildung 9.16: Anreicherung mit strukturellen Metadaten in Workflow-Client.

Google Books deutlich gemacht werden.

Online-Publikation

Die Präsentations-Plattform *Virtuelle Bibliothek Würzburg*[27] (VB) wurde eigens eingerichtet, um dem Nutzer ein originalgetreues, hochqualitatives und umfassendes Bild eines digitalisierten Werks zu verschaffen und wissenschaftliches Arbeiten auf diesen Quellen zu ermöglichen. Das vorgestellte Workflow-System kann diese Präsentations-Plattform direkt aus dem Digitalisierungs-Workflow bedienen und ein Digitalisat, welches die Produktionskette erfolgreich durchlaufen hat, automatisiert publizieren.

[27] http://vb.uni-wuerzburg.de/ub/index.html

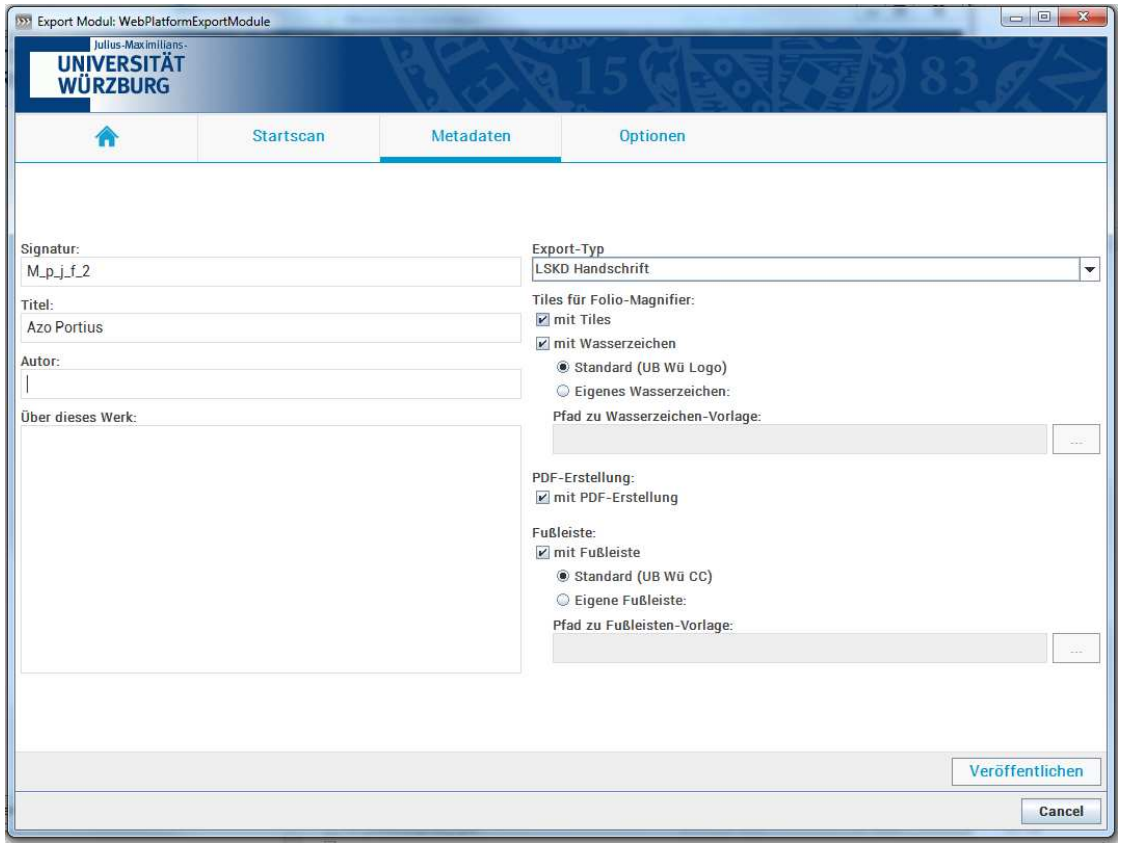


Abbildung 9.17: Export-Modul zur Veröffentlichung von Digitalisaten auf Web-Portal http://vb.uni-wuerzburg.de.

Im Gegensatz zu den in Abschnitt 9.3 angesprochenen Workflow-Systemen erlauben die Präsentations-Werkzeuge, die in der VB verwendet werden, nicht nur die Darstellung der reinen Bilddaten, sondern verwenden auch die (automatisiert) gesammelten Metadaten für unterschiedlich akzentuierte Präsentationsmodi.

Abbildung 9.18 zeigt eine klassische Detailansicht einer digitalisierten Seite eines Werks (hier und auf den nachfolgenden Abbildungen *Epistolae Pauli, M.p.th.f.68*). Die verschiedenen Bildauflösungen, die für die Navigations-Übersicht, für die zentrale Bildvorschau sowie die Detailansicht benötigt werden, generiert das Workflow-System während des Publikations-Vorgangs automatisiert. Die qualitativ hochwertige Detailansicht wird dabei mit Wasserzeichen gegen Missbrauch abgesichert. Die im Bild zu erkennende hierarchische Navigation (Inhalt, Einband, etc) wird aus

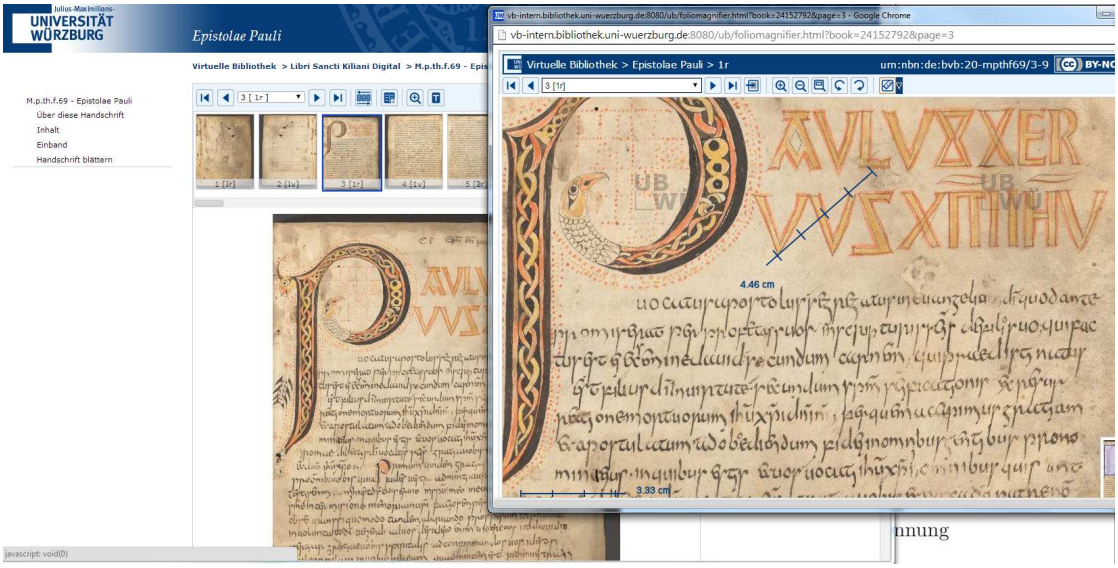


Abbildung 9.18: Detailansicht mit Lupen- und Maßband-Funktionalität. Epistolae Pauli.

strukturellen Metadaten generiert, die direkt dem Digitalisierungs-Workflow entstammen. Die Seitenabfolge der Vorschau-Ansicht kann mit dem in Abschnitt 9.4 angesprochenen Ansatz zur Inhaltsverzeichnis-Erkennung validiert werden.
Die Qualität der Präsentation sowie die strukturierte Navigation stellen einen deutlichen Kontrast zu den auf Masse und Geschwindigkeit ausgelegten Digitalisierungs-Initiativen dar (wie der Vergleich zu Abbildung 1.2 auf Seite 15 verdeutlicht).

Abbildung 9.19 zeigt eine alternative Darstellung, die dem Nutzer das Gefühl vermitteln soll, in einem echten Buch zu blättern. Dafür sind beschnittene Kopien des originalen Digitalisats mit transparentem Hintergrund notwendig. Mit Hilfe des in Abschnitt 9.4 vorgestellten Tools zur Freistellung einzelner Seiten können die dafür notwendigen Bilddaten bereits im Workflow-System generiert werden.

Abbildung 9.20 zeigt eine synoptische Gegenüberstellung von Bild- und Volltext-Daten einer einzelnen Seite aus Epistolae Pauli. Der oberste Ausschnitt präsentiert einen Bereich aus der originalen Abbildung, der mittlere Ausschnitt stellt den gleichen Bereich als gerendertes Transkript dar, während der untere Ausschnitt den Bereich romanisiert präsentiert. Dem Nutzer wird so das Lesen der originalen Abbildung erleichtert. Für diese Darstellung werden neben den Bild- auch Volltext-

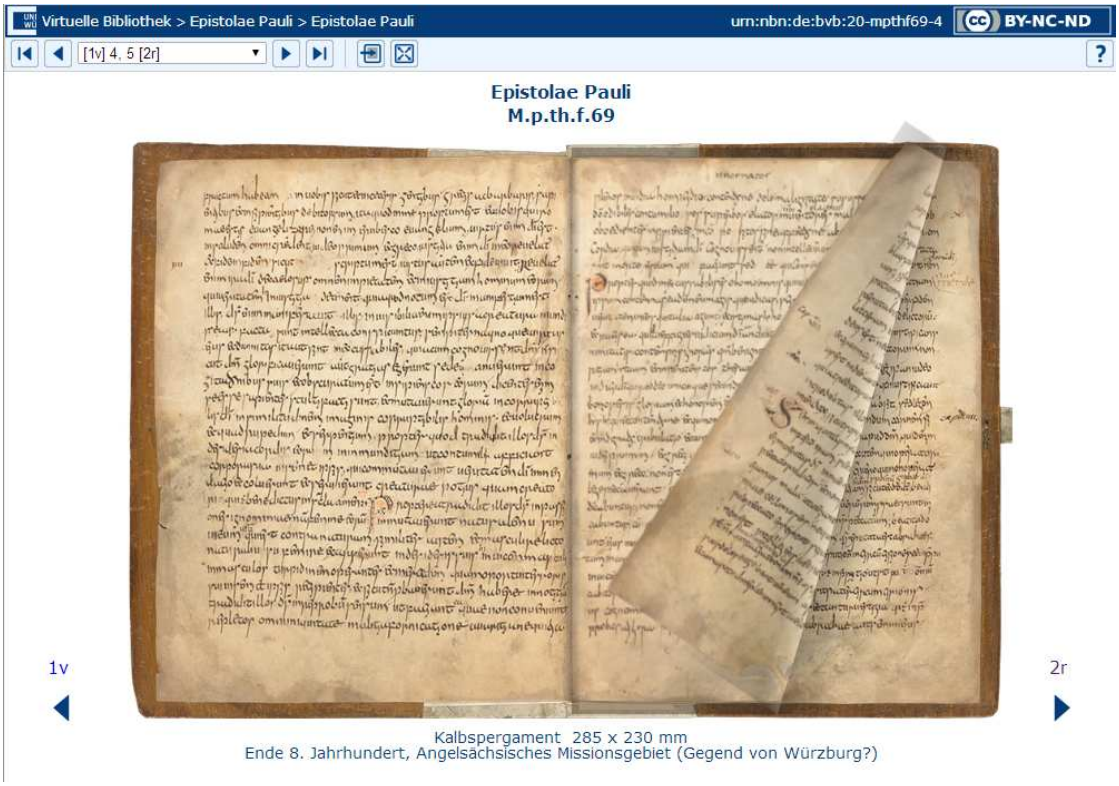


Abbildung 9.19: M.p.th.f.69 Blätteranimation.

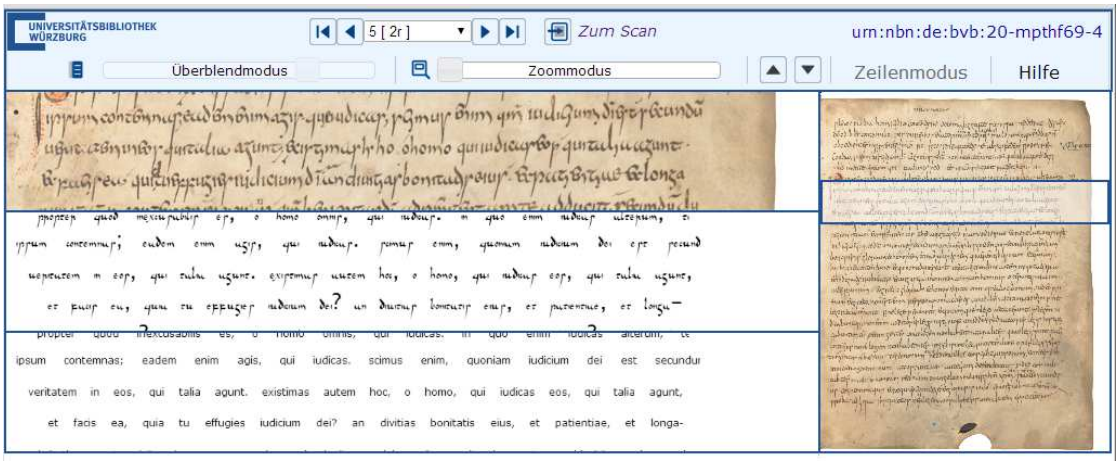


Abbildung 9.20: M.p.th.f.69 Bild- und Volltext-Überlagerung.

Daten benötigt; dem System muss außerdem eine zeilengenaue Zuordnung zwischen Bild und Volltext gegeben werden - dieser Vorgang ist bis dato allerdings

nicht automatisiert umgesetzt.

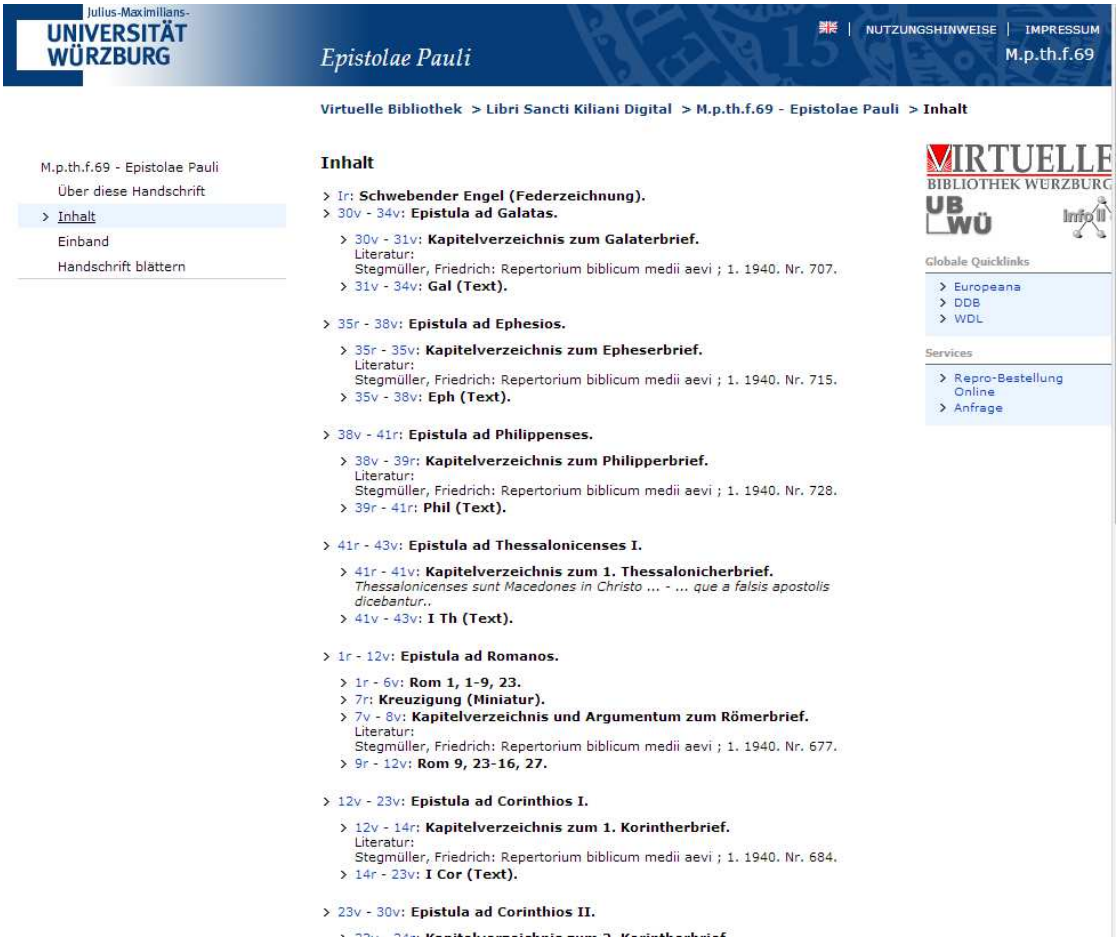


Abbildung 9.21: M.p.th.f.69 Inhalt und Metadaten.

Abbildung 9.21 konzentriert sich auf eine detailreiche Darstellung der erfassten Metadaten zu einzelnen Abschnitten innerhalb eines Werkes. Die Zuordnung zwischen Bild- und Metadaten erfolgt anhand der im Workflow-System gewonnenen strukturellen Metadaten, beispielsweise durch die Inhaltsverzeichnis-Analyse und die Seitenzahl-Plausibilisierung (besprochen in Abschnitt 9.4). Auch nach der Publikation können die Metadaten eines Digitalisats durch webgängige Werkzeuge von Besuchern der Webseite über ein moderiertes Wiki-System und Fachleuten über einen Metadaten-Editor gepflegt werden. Die Metadaten erlauben eine detaillierte Annotation der Bilddaten und Querverweise zu verwandter oder weiterführender

Literatur.

Um eine Quelle durchsuchbar und auch einzelne Passagen zitierbar machen zu können, werden der Volltext eines Digitalisats und sämtliche gesammelten Metadaten in einem Such-Index erfasst. Die Annotationen, die über die kontextuelle Klassifikation der in dieser Arbeit vorgestellten Verfahren gewonnen werden können, erlauben darüber hinaus nicht nur wortgenaue Treffer, auch unscharfe oder konzeptuelle Begriffe können damit gefunden werden.

9.7 Fazit

Das in der Digitalisierungsabteilung der Unversitätsbibliothek Würzburg eingesetzte Workflow-System hat sich als effektives Werkzeug bewährt, welches nicht nur die Durchlaufzeit von Digitalisierungsprojekten deutlich reduziert, sondern auch höchste Qualität der Digitalisate durch frühzeitige und zuverlässige Erkennung von Produktionsfehlern gewährleistet.

Weiterhin ist in Abgrenzung zu marktüblichen Verfahren die Erfassung und Pflege von Metadaten expliziter Bestandteil der Verarbeitungskette. Durch die automatisierte Extraktion von Metadaten kann das Digitalisat dem Betrachter originalgetreuer und vielfältiger präsentiert, sowie dessen Inhalt durchsuchbar gemacht und Querverweise zu weiterführenden Dokumenten hergestellt werden. Die Metadaten erlauben darüber hinaus zusätzliche Fehlerheuristiken zur Sicherung der Produktionsqualität. Insbesondere kommen dabei die in Kapitel 5 vorgestellten und weitere automatisierte Ansätze zur Metadaten-Extraktion zum Einsatz. Vorgestellt wurde auch, dass das Workflow-System auf verschiedenen Verarbeitungswegen einen bidirektionalen Zugriff auf Metadaten ermöglicht. Dadurch eröffnen sich vielfältige Kooperationsmöglichkeiten in Initiativen der Digital Humanities, da beispielsweise externe Domänen-Experten ihr Wissen einfließen oder interessierte Anwender bestehende Daten einsehen können, ohne sich mit Implementierungs-Details des Workflow-Systems auseinandersetzen zu müssen.

Die vielfältigen Export-Ziele sowie die modulare, entkoppelte Architektur machen das Workflow-System zukunftssicher und flexibel bei der Anpassung an sich verändernde Standards und gewährleisten, dass es sich bei den erzeugten Digitalisaten keinesfalls nur um spezifisch verwertbare „Insellösungen“ handelt, sondern um sorgfältig gepflegte Reproduktionen, die es für die Nachwelt in möglichst vielen Manifestationen zu erhalten gilt.

Die Verwendung des Aktoren-Modells bei der Implementierung hat nicht nur zu einer hohen Robustheit auch gegenüber unvorhergesehenen Fehlerfällen geführt, sondern ermöglicht auch die effiziente Ausnutzung im Netzwerk vorhandener Rechen-Kapazitäten, um die hohen anfallenden Belastungsspitzen abzudämpfen.
Weiterhin wurde durch die Verwendung der monadischen Future-Komposition und durch den Einsatz der Enterprise Integration Patterns zum Nachrichtenversand das Entwurfsziel erreicht, je nach Digitalisat und Verwendungszweck dynamische Ausführungspfade bilden zu können.

10 Zusammenfassung und Ausblick

Das nachfolgende Kapitel soll die in dieser Arbeit erreichten Ziele verdeutlichen. Dazu werden die Eigenschaften der vorgestellten Ansätze zusammengefasst und ihre Wirksamkeit beschrieben. Im Anschluss daran wird über mögliche Erweiterungen informiert.

10.1 Ergebnisse

In diesem Abschnitt werden die erreichten Ziele dieser Arbeit zusammengefasst und ihre Leistung bewertet.

- **Kontextuelle Klassifikation** In dieser Arbeit wurden drei Verfahren vorgestellt, die die Information aus dem Kontext eines Terms verwenden, um eine Klassifikation vorzunehmen. Da der Kontext häufig aus orthographisch konsistenteren Stopworten besteht, können auch in syntaktisch schwierigen Quellen wie historischen Texten, beschädigten Dokumenten oder fehlerhaften OCR-Ergebnissen statistisch relevante Muster gefunden werden, die für eine Klassifikation des nachfolgenden Terms verwendet werden können.
- **Domänenstabilität** Insbesondere die hybride Verwendung von statistischen und domänenspezifischen Feature-Funktionen zur Erzeugung des Eigenschaften-Vektors, anhand dessen die Klassifikation im Kontext-Vektor Ansatz vorgenommen wird, erlaubt eine stabile Klassifikationsleistung über wechselnde Domänen hinweg. Dadurch kann dieser Ansatz im Rahmen einer automatisierten Verarbeitungskette Metadaten für eine Vielzahl unterschiedlicher Quellen erzeugen.

- **Vieldeutigkeit** Durch die Verwendung der Informationen aus dem Kontext eines Terms kann situationsabhängig entschieden werden, wie doppeldeutige Terme zu klassifizieren sind. Für Verfahren, die auf Vergleiche von Zeichensequenzen setzen, ist das nicht möglich.

- **Workflow-System für Digitalisierungsprozesse** Diese Arbeit hat ein Workflow-System für die Digitalisierung historischer Dokumente vorgestellt. Durch die Verwendung automatisierter Module zur Qualitätssicherung konnte die Reproduktionsqualität maximiert und die Durchlaufzeit für Digitalisierungsprojekte drastisch gesenkt werden.

- **Explizite Behandlung von Metadaten** In Abgrenzung zu üblichen Verfahren sieht die vorgestellte Verarbeitungskette Metadaten als integralen Bestandteil eines Digitalisats vor und bietet automatisierte Ansätze zur Metadaten-Extraktion an. Die Metadaten werden einerseits als Grundlage für Fehlerheuristiken verwendet, erlauben andererseits die Darstellung des digitalisierten Werks in originalgetreuerer Form und ermöglichen eine intuitive, werksspezifische Navigation, Suche und Querverweise in weiterführende Dokumente. Weiterhin ist die ganzheitliche Verantwortlichkeit des Workflow-Systems vom Scanner bis hin zur Langzeit-Archivierung und zum Export in Präsentations- oder Metadaten-Austausch-Plattformen ein Alleinstellungsmerkmal.

- **Integration** Durch vielfältige Kommunikations-Endpunkte und ein lose gekoppeltes, modulares Design können Entwickler leicht eigene Komponenten in den Workflow einbringen oder extern gepflegte Metadaten zu einem Digitalisat hinzufügen.

10.2 Mögliche Erweiterungen

In diesem Abschnitt soll eine Übersicht über mögliche Erweiterungen der vorgestellten Ansätze gegeben werden.

- **Alternative Ähnlichkeitsmaße** Zusätzlich zum im vorgestellten Verfahren verwendeten Cosinus-Maß könnte die Leistungsfähigkeit weiterer Ähnlichkeitsmaße und Metriken für die Klassifikation von Termen in historischen Texten evaluiert werden.
- **Export-Ziele erweitern** Das vorgestellte Workflow-System ist modular erweiterbar. Vor dem Hintergrund, ein digitalisiertes Werk in mehreren redundanten Kopien für die Zukunft zu erhalten und es einem möglichst breiten Publikum zugänglich zu machen, sollte die Liste der möglichen Export-Ziele und unterstützten Metadaten-Formate erweitert werden.
- **Metadaten-Import** Bereits jetzt unterstützt das Workflow-System eine breite Palette an Kommunikations-Endpunkten, über die Bild- und Metadaten ins System eingespeist und ausgelesen werden können. Die beim Datenimport angestoßenen Verarbeitungswege können je nach Datentyp und Verwendungszweck variieren. Hier könnte die Liste der Formate und Behandlungsroutinen erweitert werden, um weitere Kooperationspartner zu gewinnen.
- **Qualitätskontrollen** Die Liste der verwendeten automatisierten Qualitätsmodule deckt viele häufig auftretende Fehlertypen ab. Sie könnte für spezielle Fehlertypen oder zukünftig hinzugenommene Objekttypen erweitert werden.
- **Metadaten-Module** Wie auch die Qualitätsmodule können die Module zur Metadaten-Extraktion erweitert werden, um spezielle Digitalisate automatisiert zur Informations-Extraktion analysieren zu können.
- **Clustering** Um die Gefahr eines *Single Point of Failures* zu reduzieren, könnte der Server statt nur auf einem einzigen Rechner in einem Cluster von Rechnerknoten operieren. Die Grundsteine dazu wurden mit der Verwendung des Aktoren-Modells gelegt.

Literatur

[1] Gene M. Amdahl. “Validity of the single processor approach to achieving large scale computing capabilities”. In: *Proceedings of the April 18-20, 1967, spring joint computer conference*. AFIPS ’67 (Spring). Atlantic City, New Jersey: ACM, 1967, S. 483–485. DOI: 10.1145/1465482.1465560. URL: http://doi.acm.org/10.1145/1465482.1465560.

[2] Galen Andrew und Jianfeng Gao. “Scalable training of L1-regularized log-linear models”. In: *Proceedings of the 24th international conference on Machine learning*. ICML ’07. Corvalis, Oregon: ACM, 2007, S. 33–40. ISBN: 978-1-59593-793-3. DOI: 10.1145/1273496.1273501. URL: http://doi.acm.org/10.1145/1273496.1273501.

[3] Paulus Apostulus. *Epistolae Pauli*. 790. URL: http://vb.uni-wuerzburg.de/ub/mpthf69/pages/mpthf69/ueber.html.

[4] A.V. Arkhangelskii und V.V. Fedorchuk. “The Basic Concepts and Constructions of General Topology”. English. In: *General Topology I*. Hrsg. von A.V. Arkhangelskii und L.S. Pontryagin. Bd. 17. Encyclopaedia of Mathematical Sciences. Springer Berlin Heidelberg, 1990, S. 1–90. ISBN: 978-3-642-64767-3. DOI: 10.1007/978-3-642-61265-7_1. URL: http://dx.doi.org/10.1007/978-3-642-61265-7_1.

[5] Ricardo A. Baeza-Yates und Berthier A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999. ISBN: 0-201-39829-X. URL: http://www.ischool.berkeley.edu/~hearst/irbook/glossary.html.

[6] Henry C. Baker Jr. und Carl Hewitt. “The incremental garbage collection of processes”. In: *Proceedings of the 1977 symposium on Artificial intelligence and programming languages*. New York, NY, USA: ACM, 1977, S. 55–59. DOI: 10.1145/800228.806932. URL: http://doi.acm.org/10.1145/800228.806932.

[7] Simon Beretta. “Hierarchisches OCR - Heuristiken zur partiellen Erschließung von Retrodigitalisaten.” Diplomarbeit. Julius-Maximilians-Universität Würzburg, 2012.

[8] Kevin Beyer u. a. “When Is "Nearest Neighbor"Meaningful?” In: *Int. Conf. on Database Theory*. 1999, S. 217–235.

[9] Holger Billhardt u. a. “A Context Vector Model for Information Retrieval”. In: *Journal of the American Society for Information Science and Technology* 53 (2002), S. 236–249.

[10] Steven Bird, Ewan Klein und Edward Loper. *Natural language processing with Python*. O’Reilly, 2009.

[11] Marcel Bollmann, Florian Petran und Stefanie Dipper. “Rule-based normalization of historical texts”. In: *Proceedings of the International Workshop on Language Technologies for Digital Humanities and Cultural Heritage*. 2011, S. 34–42.

[12] K. Byrne. “Nested Named Entity Recognition in Historical Archive Text”. In: *International Conference on Semantic Computing, 2007. ICSC 2007.* 2007, S. 589–596. DOI: 10.1109/ICSC.2007.107.

[13] William B. Cavnar und John M. Trenkle. “N-gram based text categorization”. In: *In Proc. of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*. 1994, S. 161–175.

[14] LJC Book Club. *Programming Concurrency on the JVM*. Feb. 2014. URL: http://ljcbookclub.wordpress.com/author/twr0bel/.

[15] Marco Dittrich und Ulf Weinmann. *Persönliche Kommunikation*. 2014.

[16] Kevin Dooley. *Designing large scale LANS*. O’Reilly Media, Inc., 2001.

[17] J. Stephen Downie u. a., Hrsg. *13th ACM/IEEE-CS Joint Conference on Digital Libraries, JCDL '13, Indianapolis, IN, USA - July 22 - 26, 2013*. ACM, 2013. ISBN: 978-1-4503-2077-1.

[18] *Duden*. 2014. URL: http://www.duden.de/rechtschreibung/digitalisieren.

[19] Ted Dunning. *Statistical Identification of Language*. Techn. Ber. 1994.

[20] R. Durrett. *Probability: Theory and Examples*. 4th Edition. Cambridge U Press, 2010.

[21] Laurene Fausett. *Fundamentals of neural networks: architectures, algorithms, and applications*. Prentice-Hall, Inc., 1994.

[22] Reginald Ferber. *Information Retrieval: Suchmodelle und Data-Mining-Verfahren für Textsammlungen und das Web*. Heidelberg: dpunkt Verlag, 2003. URL: http://information-retrieval.de/.

[23] Roy Thomas Fielding. "REST: Architectural Styles and the Design of Network-based Software Architectures". Doctoral dissertation. University of California, Irvine, 2000. URL: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

[24] Jr. Forney G.D. "The Viterbi algorithm". In: *Proceedings of the IEEE* 61.3 (1973), S. 268–278. ISSN: 0018-9219. DOI: 10.1109/PROC.1973.9030.

[25] Martin Fowler. *Inversion of Control Containers and the Dependency Injection pattern*. Jan. 2004. URL: http://martinfowler.com/articles/injection.html.

[26] *Frankfurter Rundschau*. 2013. URL: http://www.fr-online.de/medien/google-books-googles-kampf-um-die-buecher,1473342,22255046.html.

[27] Yoav Freund und Robert E. Schapire. *A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting*. 1997.

[28] fxjwind. *Akka Essentials*. 2014. URL: http://www.cnblogs.com/fxjwind/p/3428258.html.

[29] Susan Gauch, Jianying Wang und Satya Mahesh Rachakonda. “A corpus analysis approach for automatic query expansion and its extension to multiple databases”. In: *ACM Trans. Inf. Syst.* 17.3 (1999), S. 250–269. ISSN: 1046-8188. DOI: http://doi.acm.org/10.1145/314516.314519.

[30] *GeoNames*. 2014. URL: http://download.geonames.org/export/dump/.

[31] V.N. Grishin. *Encyclopedia of Mathematics - Tuple*. 2011. URL: http://www.encyclopediaofmath.org/index.php?title=Tuple&oldid=11398.

[32] Philipp Haller u. a. *Futures and Promises*. 2013. URL: http://docs.scala-lang.org/sips/pending/futures-promises.html.

[33] Andreas Hauser u. a. “Information Access to Historical Documents from the Early New High German Period”. In: *Digital Historical Corpora- Architecture, Annotation, and Retrieval*. Hrsg. von Lou Burnard u. a. Dagstuhl Seminar Proceedings 06491. Dagstuhl, Germany: Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2007. URL: http://drops.dagstuhl.de/opus/volltexte/2007/1057.

[34] *Heise online*. 2013. URL: http://www.heise.de/newsticker/meldung/Google-Books-nach-acht-Jahren-vor-Gericht-fuer-legal-erklaert-2046814.html.

[35] Winfried Höhn, Hans-Günter Schmidt und Hendrik Schöneberg. “Semiautomatic recognition and georeferencing of places in early maps”. In: *JCDL*. Hrsg. von J. Stephen Downie u. a. ACM, 2013, S. 335–338. ISBN: 978-1-4503-2077-1.

[36] Gregor Hohpe und Bobby Woolf. *Enterprise Integration Patterns: Designing, Building and Deploying Messaging Solutions*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003. ISBN: 0321200683.

[37] Michael E. Houle u. a. “Can Shared-Neighbor Distances Defeat the Curse of Dimensionality?” In: *Scientific and Statistical Database Management*. Hrsg. von Michael Gertz und Bertram Ludäscher. Bd. 6187. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, S. 482–500. ISBN: 978-3-642-13817-1. DOI: 10.1007/978-3-642-13818-8_34. URL: http://dx.doi.org/10.1007/978-3-642-13818-8_34.

[38] http://akka.io. *Supervision and Monitoring*. Aug. 2013. URL: http://doc.akka.io/docs/akka/2.1.4/general/supervision.html.

[39] Text Encoding Initiative. *TEI: P5 Guidelines*. URL: http://www.tei-c.org/Guidelines/P5/.

[40] Mark P. Jones und Luc Duponcheel. *Composing Monads*. Techn. Ber. 1993.

[41] Woo Young Kim und Gul Agha. "Efficient support of location transparency in concurrent object-oriented programming languages". In: *Supercomputing, 1995. Proceedings of the IEEE/ACM SC95 Conference*. IEEE. 1995, S. 39–39.

[42] Peter Kluegl u. a. "Collective Information Extraction with Context-Specific Consistencies". In: *Machine Learning and Knowledge Discovery in Databases*. Hrsg. von PeterA. Flach, Tijl Bie und Nello Cristianini. Bd. 7523. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, S. 728–743. ISBN: 978-3-642-33459-7. DOI: 10.1007/978-3-642-33460-3_52. URL: http://dx.doi.org/10.1007/978-3-642-33460-3_52.

[43] Peter Kluegl u. a. "Exploiting Structural Consistencies with Stacked Conditional Random Fields". In: *Mathematical Methodologies in Pattern Recognition and Machine Learning Springer Proceedings in Mathematics Statistics* 30 (2013), S. 111–125.

[44] John Lafferty, Andrew McCallum und Fernando Pereira. "Conditional random fields: Probabilistic models for segmenting and labeling sequence data". In: *Proceedings of the 18th International Conference on Machine Learning*. San Fransisco: Morgan Kaufmann, 2001, S. 282–289.

[45] *LePublikateur*. 2013. URL: http://www.lepublikateur.de/2013/11/17/google-books-ist-der-feind-und-wissen-ist-macht/.

[46] *Mallet Optimization*. 2014. URL: http://mallet.cs.umass.edu/optimization.php.

[47] C. D. Manning, P. Raghavan und H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. URL: http://www-csli.stanford.edu/~hinrich/information-retrieval-book.html.

[48] Christopher D. Manning und Hinrich Schütze. *Foundations of statistical natural language processing*. Bd. 999. MIT Press, 1999.

[49] E. Marsh und D. Perzanowski. "MUC-7 Evaluation of IE Technology: Overview of Results". In: *Proceedings of the Seventh Message Understanding Conference (MUC-7)*. 1998.

[50] Matthäus der Ältere Merian. *Topographia Germaniae*. 1642.

[51] M. Michael u. a. "Scale-up x Scale-out: A Case Study using Nutch/Lucene". In: *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*. 2007, S. 1–8. DOI: 10.1109/IPDPS.2007.370631.

[52] George A. Miller und Walter G. Charles. "Contextual correlates of semantic similarity". In: *Language and Cognitive Processes* 6 (1991).

[53] Frank Müller. "Identifikation von Toponymen in historischen Texten". Diplomarbeit. Julius-Maximilians-Universität Würzburg, Institut für Informatik, 2012.

[54] Deutsche Nationalbibliothek. *Gemeinsame Normdatei (GND)*. 2014. URL: http://www.dnb.de/gnd.

[55] *ocropus*. URL: https://code.google.com/p/ocropus/.

[56] Oracle. *Oracle ServiceLoader<S> API Dokumentation*. URL: http://docs.oracle.com/javase/7/docs/api/java/util/ServiceLoader.html.

[57] Oracle. *Package java.util.stream*. 2014. URL: http://docs.oracle.com/javase/8/docs/api/java/util/stream/package-summary.html.

[58] National Information Standards Organization. *Understanding Metadata*. NISO Press. Bethesda, MD 20814 USA, 2004.

[59] Michael Piotrowski. *Natural Language Processing for Historical Texts*. Synthesis Lectures on Human Language Technologies. Morgan Claypool Publishers, 2012. URL: http://dx.doi.org/10.2200/S00436ED1V01Y201207HLT017.

[60] Thierry Poibeau und Leila Kosseim. "Proper Name Extraction from Non-Journalistic Texts". In: *Computational Linguistics in the Netherlands*. 2001, S. 144–157.

[61] U. Quasthoff, M. Richter und C. Biemann. “Corpus Portal for Search in Monolingual Corpora”. In: *Proceedings of the fifth international conference on Language Resources and Evaluation, LREC*. Genoa, 2006, S. 1799–1802.

[62] Lawrence R Rabiner. “A tutorial on hidden Markov models and selected applications in speech recognition”. In: *Proceedings of the IEEE* 77.2 (1989), S. 257–286.

[63] Dirk Riehle. “Role model based framework design and integration”. In: *Proceedings of the 1998 Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA 98)*. ACM Press, 1998, S. 117–133.

[64] Kepa Joseba Rodriquez u. a. “Comparison of named entity recognition tools for raw OCR text”. In: *Proceedings of KONVENS*. 2012, S. 410–414.

[65] Frank Rosenblatt. “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain”. In: *Psychological Review* 65.6 (1958), S. 386–408.

[66] G. Salton, A. Wong und C. S. Yang. “A Vector Space Model for Automatic Indexing”. In: *Commun. ACM* 18.11 (Nov. 1975), S. 613–620. ISSN: 0001-0782. DOI: 10.1145/361219.361220. URL: http://doi.acm.org/10.1145/361219.361220.

[67] Hans-Günter Schmidt. *Persönliche Kommunikation*. 2014.

[68] Peter Schneider. “Leistungsvergleiche zu Verfahren der Toponym-Erkennung in historischen Texten”. Bachelorarbeit. Julius-Maximilians-Universität Würzburg, Institut für Informatik, 2013.

[69] Hendrik Schöneberg. “Context Vector Classification - Term Classification with Context Evaluation”. In: *KDIR*. Hrsg. von Ana L. N. Fred und Joaquim Filipe. SciTePress, 2010, S. 387–391. ISBN: 978-989-8425-28-7.

[70] Hendrik Schöneberg und Frank Müller. “Contextual Approaches for Identification of Toponyms in Ancient Documents”. In: *KDIR*. Hrsg. von Ana L. N. Fred u. a. SciTePress, 2012, S. 163–168. ISBN: 978-989-8565-29-7.

[71] Hendrik Schöneberg, Hans-Günter Schmidt und Winfried Höhn. “A scalable, distributed and dynamic workflow system for digitization processes”. In: *JCDL*. Hrsg. von J. Stephen Downie u. a. ACM, 2013, S. 359–362. ISBN: 978-1-4503-2077-1.

[72] H. Schütze. “Dimensions of meaning”. In: *Supercomputing ’92: Proceedings of the 1992 ACM/IEEE conference on Supercomputing*. Minneapolis, Minnesota, United States: IEEE Computer Society Press, 1992, S. 787–796. ISBN: 0-8186-2630-5.

[73] David A. Smith und Gregory Crane. “Disambiguating Geographic Names in a Historical Digital Library”. In: *In Proceedings of ECDL*. Springer-Verlag, 2001, S. 127–136.

[74] Jan A. Snyman. *Practical mathematical optimization : an introduction to basic optimization theory and classical and new gradient-based algorithms*. Applied optimization. New York: Springer, 2005. ISBN: 0-387-24348-8. URL: http://opac.inria.fr/record=b1132592.

[75] springsource.com. *Java-based container configuration*. Jan. 2013. URL: http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/beans.html#beans-java.

[76] *Tesseract OCR*. URL: https://code.google.com/p/tesseract-ocr/.

[77] *The Reactive Manifesto*. 2014. URL: http://www.reactivemanifesto.org/.

[78] *Tiff 6.0 Image Specification*. URL: http://partners.adobe.com/public/developer/tiff/index.html.

[79] Emmanuel Donfack Tiona. “Effizienz und Fehlertoleranz im Workflow zur Retrodigitalisierung alter Handschriften und Drucke”. Diplomarbeit. Julius-Maximilians-Universität Würzburg, 2013.

[80] *TreeTagger*. 2014. URL: http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/.

[81] Typesafe. *Akka Futures*. 2013. URL: http://doc.akka.io/docs/akka/2.1.4/java/futures.html.

[82] *wirtschaftslexikon24.com*. Jan. 2014. URL: http://www.wirtschaftslexikon24.com/d/durchlaufzeit/durchlaufzeit.htm.

[83] Ian H. Witten, Eibe Frank und Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. 3rd. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011. ISBN: 0123748569, 9780123748560.

[84] Universitätsbibliothek Würzburg. *Libri Sancti Kiliani digital*. Jan. 2014. URL: http://www.libri-kiliani.eu/.

[85] Universitätsbibliothek Würzburg. *Virtuelle Bibliothek Würzburg*. Jan. 2014. URL: http://vb.uni-wuerzburg.de/.

[86] Derek Wyatt. *Balancing Workload Across Nodes with Akka 2*. 2013. URL: http://letitcrash.com/post/29044669086/balancing-workload-across-nodes-with-akka-2/.