Regelbasierte Informationsextraktion aus strukturierten Dokumenten

Günter Vajde

MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im Januar 2013

Erklärung

Ich erkläre eidesstattlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen entnommenen Stellen als solche gekennzeichnet habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Hagenberg, am 25. Januar 2013

Günter Vajde

Inhaltsverzeichnis

Kurzfassung

Diese Arbeit beschäftigt sich mit der Informationsextraktion aus strukturierten Dokumenten. Die Thematik wird anhand der Erstellung von Intrastatmeldungen auf Basis von gescannten Rechnungen beleuchtet. Es wird gezeigt, wie die Ausgangsdaten schrittweise in maschinenlesbare und für den Benutzer bedeutungsvolle Informationen transformiert werden können.

Die Informationsextraktion erfolgt maßgeblich auf Basis von Layoutstrukturen und Beziehungen zwischen Schlüsselwörtern und Datenfeldern. Der Ansatz eignet sich daher besonders um Informationen aus Tabellen zu extrahieren, bei denen die Bedeutung von Daten aufgrund ihrer relativen Position zu bestimmten Überschriften hervorgeht. Repräsentiert wird das Extraktionswissen durch eine Menge von klassenspezifischen Regeln, welche durch den Reasoner des RDF-Frameworks *Jena* angewandt werden. Dabei wird darauf eingegangen, welche Maßnahmen zur Optimierung der Performance des Reasoners getroffen werden können.

Abschließend wird das entwickelte System evaluiert. Dabei werden die Ergebnisse der Informationextraktion vorgestellt, Fehlerquellen identifiziert und weiteres Verbesserungspotential thematisiert.

Abstract

This thesis deals with information extraction from structured documents. The subject is explained by showing how to convert invoices into Intrastat declarations. For this purpose, the initial data are gradually being transformed into machine-readable information that are also meaningful to a human operator.

Information are extracted by employing physical layout features and relationships between keywords and data fields. Therefore, the presented approach is specifically suited for extracting information from tables, as the meaning of its contained data can usually be inferred by associated headers. Extraction knowledge is represented by a set of class-specific rules, that are applied on single pages using Jena's rule engine. In this regard, it is shown which measures can be taken to improve reasoning performance.

The thesis concludes by evaluating the presented approach. The quality of the extracted information is analyzed, sources of errors are identified and possible future enhancements are discussed.

Kapitel 1

Einleitung

Im Zuge von Prozessoptimierungen wurden in den letzten Jahren Bürotätigkeiten zunehmend automatisiert. Eine dieser Tätigkeiten betrifft die Verarbeitung von Dokumenten, deren Automatisierung, mit Marktreife der Texterkennungstechnologie (engl. *Optical Character Recognition*, OCR), besonders seit Ende der 1990er-Jahre stetig vorangetrieben wurde. Aufgrund der Leistungsfähigkeit heutiger Computersysteme und der Verfügbarkeit hochauflösender Dokumentenscanner, sind die technischen Grundvoraussetzungen für den Betrieb von Dokumentenverarbeitungssystemen auch für Kleinunternehmen mit überschaubaren Investitionskosten herzustellen.

Da der Bearbeitung von strukturierten Geschäftsdokumenten, wie Rechnungen, formal definierbare Kriterien zugrunde liegen, bieten sich diese – im Gegensatz zu textlastigen Dokumenten, die einer Interpretation des Inhaltes bedürfen – besonders zur Automatisierung an. In vielen Anwendungsfällen besteht die Bearbeitung dieser Dokumente daraus, dass bestimmte Daten ausgelesen und dem Geschäftsprozess entsprechend klassifiziert werden. Die klassifizierten Daten werden schließlich in einem bekannten Zielsystem manuell erfasst. Die manuelle Bearbeitung durch einen Experten erfolgt durch Anwendung seines Fachwissens und seiner Erfahrung. Mithilfe semantischer Technologien ist es möglich das, oft nur implizit bestehende, Expertenwissen nachzumodellieren.

Da es weiterhin üblich ist, Geschäftsdokumente physisch über den Postweg auszutauschen und deren manuelle Bearbeitung bei großen Mengen mit beträchtlichem Zeitaufwand verbunden ist, wird im Rahmen dieser Arbeit ein Ansatz zur Informationsextraktion aus strukturierten Dokumenten vorgestellt. Als konkreten Anwendungsfall wurde die Erstellung von Intrastat-Meldungen auf Basis von Rechnungen gewählt. Die zugrundeliegenden Methoden lassen sich jedoch allgemein auf strukturierte Dokumente oder auch statische Formulare anwenden.

Aufbau der Arbeit

In Kapitel 2 erfolgt eine Beschreibung der wirtschaftlichen und rechtlichen Grundlagen. Es wird erklärt wofür Intrastat-Meldungen benötigt werden, woraus diese bestehen und auf welcher Basis diese erstellt werden. Die zum Verständnis der Arbeit nötigen technischen Grundlagen werden danach in Kapitel 3 vorgestellt. Dabei wird das Konzept semantischer Technologien und regelbasierter Systeme vermittelt. Kapitel 4 befasst sich mit verwandten Arbeiten aus dem Umfeld der Dokumentenverarbeitung. Exemplarisch werden einige Ansätze vorgestellt, die sich konkret mit der Verarbeitung von Rechnungen befassen und daher einige Parallelen zum entwickelten System aufweisen. Eine Übersicht des Systemaufbaus, der eingesetzten Technologien und der herangezogenen Datenquellen wird in Kapitel 5 gegeben. Kapitel 6 widmet sich den Implementierungsdetails des entwicklten Ansatzes zur Informationsextrakion. Beginnend bei der Konvertierung von Bilddateien in ein maschinenlesbares Zwischenformat, werden die einzelnen Verarbeitungsschritte bis zum Datenexport näher beschrieben. Ergebnisse der Evaluierung werden in Kapitel 7 vorgestellt. Es wurde analysiert wie zuverlässig Informationen aus Dokumenten extrahiert werden und in welchen Bereichen Verbesserungspotential besteht.

Kapitel 2

Wirtschaftliche und rechtliche Grundlagen

2.1 Außenhandelsstatistik

Die Außenhandelsstatistik [7] stellt eine wesentliche Grundlage zur Bewertung der wirtschaftlichen Situation eines Landes dar und wird unter anderem für wirtschaftspolitische Entscheidungen oder Marktanalysen verwendet. Zur Erstellung der österreichischen Außenhandelsstatistik werden monatlich Daten über Im- und Exporte beweglicher Güter erfasst und an die Statistik Austria gemeldet.

Seit dem Beitritt Österreichs zur Europäischen Union (EU) im Jahr 1995 erfolgt die Erfassung dieser Daten über die Systeme Intrastat und Extrastat. Im Handel mit Drittstaaten kommt das Extrastat-System zur Anwendung und der Warenfluss wird, wie auch vor dem EU-Betritt, anhand von Zolldeklarationen durch die Zollverwaltungsbehörden dokumentiert. Aufgrund des freien Warenverkehrs und der Abschaffung der Zölle innerhalb der EU, wurde das Intrastat-System eingeführt. Dieses sieht die Datenerhebung, bei innergemeinschaftlichen Lieferungen, direkt bei den Unternehmen vor.

Ein Unternehmen gilt für die jeweilige Handelsrichtung (Eingang oder Versendung) als meldepflichtig, wenn die Summe aller Warentransaktionen einer Richtung 500.000 Euro pro Jahr überschreitet. Sofern dieser Wert im Vorjahr nicht überschritten wurde, gilt die Meldepflicht im laufenden Jahr ab dem Monat, in dem die Überschreitung erfolgt. Diese Schwelle wurde festgelegt um kleinere Unternehmen zu entlasten, aber dennoch 95% aller Eingänge und 97% aller Versendungen, den Mindestabdeckungsgraden laut Verordnung (EG) Nr. 222/2009 [44], in die Statistik aufnehmen zu können. Im Berichtsjahr 2010 waren dadurch ca. 13.000 österreichische Unternehmen meldepflichtig [3].

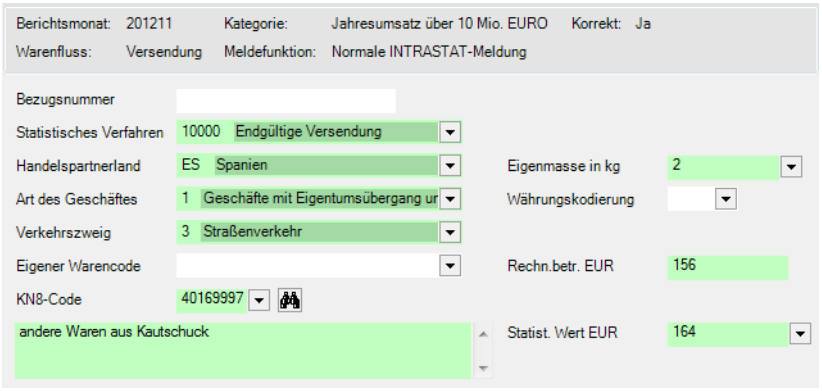


Abbildung 2.1: Erstellung einer Meldezeile in IDEP.

2.2 Intrastat-Meldung

Die Informationen, die im Rahmen des Intrastat-Systems erhoben werden, wurden durch die EU-Verordnung Nr. 638/2004 [45] festgelegt. Neben bestimmten Minimalkriterien, die für alle EU-Mitgliedsstaaten gelten, können von den nationalen Behörden weitere Informationen erfasst werden. In diesem Kapitel werden die für Österreich geltenden Bestimmungen laut Statistik Austria [4] vorgestellt. Zur Datenerfassung stehen ein Web-Formular, die Software „IDEP/KN8“ und ein Papierformular zur Verfügung. In Abbildung 2.1 ist zu sehen, wie mithilfe von IDEP ein Datensatz einer Versandmeldung erstellt wird.

Auskunftspflichtiger

Firma, Adresse und Umsatzsteueridentifikationsnummer (UID-Nummer) des auskunftspflichtigen Unternehmens werden erfasst.

Warenflussrichtung

Hinsichtlich der Warenflussrichtung wird zwischen Eingängen und Versendungen unterschieden.

Berichtszeitraum

Üblicherweise ist als Berichtszeitraum der Monat, in dem die Warenlieferung statt gefunden hat, anzugeben. Erfolgt die Ausstellung der Rechnung allerdings erst nach der Lieferung, ist diese spätestens im folgenden Monat zu melden.

Statistisches Verfahren

Die Angabe des statistischen Verfahrens dient der Unterscheidung verschiedener Arten von Warentransfers [45]. Prinzipiell wird zwischen endgültigen und vorübergehenden Lieferungen unterschieden, wodurch folgende Verfahren zur Auswahl stehen [4]:

- 10000: Endgültige Versendung,
- 22002: Vorübergehende Versendung zur wirtschaftlichen Lohnveredelung,
- 31514: Wiederversendung nach wirtschaftlicher Lohnveredelung,
- 40000: Endgültiger Eingang,
- 51004: Vorübergehender Eingang zur wirtschaftlichen Lohnveredelung,
- 61215: Wiedereingang nach wirtschaftlicher Lohnveredelung.

Eine Lohnveredelung erfolgt, wenn eine Ware vorübergehend ins Ausland gebracht wird, um dort weiterverarbeitet oder verbessert zu werden. Ein Eigentumstransfer findet daher nicht statt. Den Regelfall stellen jedoch endgültige Versendungen bzw. Eingänge dar.

Handelspartnerland

Das Land, aus dem die Ware geliefert wird, ist als Handelspartnerland anzugeben. Zur Codierung wird, ebenso wie für das Ursprungsland, der ISO-Standard 3166-Alpha-2 verwendet [7].

Ursprungsland

Das Ursprungsland einer Ware wird nur bei Eingängen erfasst und ist das Land, in dem die Ware hergestellt oder zuletzt wesentlich verändert wurde. Wenn diese Information nicht bekannt ist oder die Ware aus Österreich stammt, entspricht das Ursprungsland dem Handelspartnerland. Im Gegensatz zum Handelspartnerland muss das Ursprungsland nicht unbedingt Teil der EU sein.

Art des Geschäfts

Hinsichtlich der Art des Geschäfts wird zwischen folgenden Varianten unterschieden:

- 1: Geschäfte mit Eigentumsübergang und Gegenleistung,
- 2: Rücksendung und Ersatzlieferungen,
- 3: Geschäfte mit Eigentumsübertragung ohne Gegenleistung,
- 4: Warensendungen zur Lohnveredelung oder Reparatur,
- 5: Warensendungen nach Lohnveredelung oder Reparatur,
- 6: Warenverkehr ohne Eigentumsübergang,

- 7: Warenverkehr im Rahmen gemeinsamen von Verteidigungsprogrammen oder anderen zwischenstaatlichen Programmen,
- 8: Lieferung von Baumaterial und Ausrüstungen im Rahmen von Bau- bzw. Anlagebauarbeiten als Teil eines Generalkontraktes,
- 9: Andere Geschäfte nicht anderweitig erfasst.

Verkehrszweig

Als Verkehrszweig wird das Transportmittel, mit dem eine Ware in das bzw. aus dem Land befördert wird, erfasst. Folgende Kategorien stehen dabei zur Auswahl:

- 2: Eisenbahnverkehr,
- 3: Straßenverkehr,
- 4: Luftverkehr,
- 5: Postverkehr,
- 7: Festinstallierte Transporteinrichtungen,
- 8: Binnenschiffsverkehr,
- 9: Eigenantrieb.

Warenklassifikation

Zur Warenklassifikation werden die achtstelligen Codes der Kombinierten Nomenklatur (KN) [6] verwendet. Die KN ist ein hierarchisch, in meherere Ebenen, gegliedertes Warenverzeichnis bestehend aus 98 Kapiteln und rund 10000 Zolltarifnummern [7]. Das Verzeichnis wird jährlich – zuletzt 2012 [42] – von der EU aktualisiert, um neue technologische Entwicklungen zu berücksichtigen.

Warenbezeichnung

Neben der Warennummer wird außerdem eine handelsübliche Warenbezeichnung erfasst. Anhand dieser Beschreibung muss eine eindeutige Zuordnung zu einer Zolltarifnummer möglich sein.

Eigenmasse

Die Eigenmasse der Ware wird üblicherweise in Kilogramm angeben. Ausnahmen bilden hierbei Zolltarifnummern, für die besondere Maßeinheiten, wie z.B. Stück oder Liter, definiert wurden.

Rechnungsbetrag

Als Rechnungsbetrag ist der Nettowert der Ware in Euro anzugeben. Zur Umrechnung von Fremdwährungskursen können die aktuellen Devisen- oder

Zollwertkurse verwendet werden. Bei Rücksendungen, Eingängen zur Lohnveredelung oder nicht berechneten Ersatzlieferungen ist als Betrag der Wert 0 anzugeben. Weiters sind für Wiedereingänge nach Lohnveredelung nur die, durch die Veredelung, entstandenen Kosten anzugeben.

Statistischer Wert

Der statistische Wert wird als Wert einer Ware beim Grenzübergang definiert [7] und

> ergibt sich aus dem Rechnungsbetrag der Ware, dem je nach Lieferbedingung die Fracht- und sonstigen Kosten hinzugerechnet oder abgezogen werden.

Die Software IDEP stellt zur Berechnung des statistischen Werts vordefinierte Multiplikatoren für die Lieferbedingungen „frei Haus“, „frei Grenze“ und „ab Werk“ bereit. Der statistische Wert entspricht dabei 95, 100 bzw. 105% des angegebenen Rechnungsbetrags.

2.3 Bestandteile einer Rechnung

Die Erstellung von Intrastat-Meldungen erfolgt auf Basis von Geschäftsdokumenten, die einen Warenfluss dokumentieren. Im häufigsten Fall handelt es sich dabei um Rechnungen. In diesem Kapitel werden daher die gesetzlich vorgeschriebenen Rechnungsbestandteile laut EU-Recht vorgestellt. Da ausschließlich innerhalb der EU stattfindende Warentransfers für das Intrastat-System relevant sind, können diese Bestandteile in jeder Rechnung erwartet werden. Da die gesetzlichen Vorschriften nur die Mindestkriterien definieren und in Rechnungen üblicherweise ein Fülle weiterer Informationen enthalten ist, wurden im Rahmen einer Stichprobenanalyse die üblichen Bestandteile einer Rechnung ermittelt.

2.3.1 Gesetzliche Bestandteile

Die Vorschriften zur Rechnungsstellung wurden durch die Richtlinie 2006/112/EG [43] EU-weit vereinheitlicht. Dadurch müssen Unternehmen, die im europäischen Binnenmarkt tätig sind, bei der Ausstellung von Rechnungen nationale Besonderheiten des Handelspartnerlandes nicht mehr berücksichtigen. Die folgenden Angaben gelten als Pflichtbestandteile einer Rechnung:

- Ausstellungsdatum,
- Eindeutige fortlaufende Rechnungsnummer,
- Name und Anschrift des Ausstellers,
- Name und Anschrift des Empfängers,
- UID-Nummer des liefernden Unternehmens,

- UID-Nummer des Empfängers,
- Lieferdatum,
- Menge und Bezeichnung der gelieferten Waren,
- Warenpreis pro Einheit exklusive Steuern und Abzügen,
- Steuerbeträge je Steuersatz.

2.3.2 Übliche Bestandteile

Im Rahmen einer Strichprobenanalyse wurde untersucht, welche Daten üblicherweise in Rechnungen enthalten sind. Insgesamt wurden dafür Rechnungen elf unterschiedlicher Unternehmen aus vier Ländern untersucht. Im Hinblick auf die Erstellung von Intrastat-Meldungen können die Bestandteile einer Rechnung prinzipiell in drei Kategorien eingeteilt werden:

- Intrastat-relevante Daten,
- Verwaltungsdaten und
- Irrelevante Daten.

Die Häufigkeit, mit der einzelne Bestandteile anzutreffen waren, wird im Folgenden als Prozentwert in Klammern angegeben.

Intrastat-relevante Daten

Intrastat-relevanten Daten können wiederum in diese Kategorien eingeteilt werden:

- Direkt verwendbare Daten,
- Eindeutig ableitbare Daten,
- Klassifizierungsbedürftige Daten,
- Hilfsdaten,
- Numerische Daten und
- Redundante Daten.

Direkt verwendbare Daten können unverändert übernommen werden. Im Rahmen der Stichprobenanalyse konnte als einzige direkt verwendbare Information die Zolltarifnummer laut KN ermittelt werden. Diese ist jedoch nur in 45% der Rechnungen enthalten. Daneben gibt es Daten aus denen Intrastat-Daten *eindeutig ableitbar* sind. Dazu zählen die folgenden Angaben:

- UID-Nummer des Ausstellers (100%),
- UID-Nummer des Empfängers (100%),
- Rechnungsdatum (100%) und
- Lieferdatum (82%).

Zuvor wurde erwähnt, dass eine Intrastat-Meldung auch Firma und Adresse des auskunftspflichtigen Unternehmens enthält. Diese Daten werden hier

nicht angeführt, da davon auszugehen ist, dass diese Daten intern bereits erfasst wurden und mithilfe der UID-Nummer ermittelt werden können. Ist bekannt für welches Unternehmen eine Intrastat-Meldung erstellt wird, können aufgrund der UID-Nummern die Warenflussrichtung und das Handelspartnerland abgeleitet werden. Letzteres ist möglich, da die ersten zwei Zeichen einer UID-Nummer dem jeweiligen ISO-Ländercode entsprechen [32].

Klassifizierungsbedürftige Daten können nicht ohne vorherige Klassifizierung für eine Intrastat-Meldung verwendet werden. Zu diesen Daten zählen:

- Artikelbezeichnung (100%),
- Ursprungsland (18%) und
- Versandart (55%).

Die Artikelbezeichnung hilft dabei Waren von Dienstleistungen, die nicht gemeldet werden, zu unterscheiden. Weiters wird aufgrund der Artikelbezeichnung die handelsübliche Warenbezeichnung bestimmt und, sofern keine entsprechende Angabe vorhanden ist, die Ware laut KN klassifiziert. Das Ursprungsland wurde in 18% der Rechnungen angegeben, jedoch in diesen Fällen immer in Klartext. Somit ist eine direkte Verwendung nicht gegeben, da zuerst der entsprechende ISO-Ländercode ermittelt werden muss. Die Versandart wurde in 55% der Fälle angegeben und bestimmt den Verkehrszweig.

Hilfsdaten sind Daten, die Hinweise zur Um- oder Berechnung eines numerischen Werts bieten, in einer Intrastat-Meldung jedoch nicht direkt aufscheinen. Folgende Daten zählen zu dieser Kategorie:

- Währung (100%),
- Lieferbedingungen (82%),
- Mengeneinheit pro Position (91%),
- Gewichtseinheit pro Position (9%),
- Gesamtgewicht der Lieferung (91%).

Da in den wenigsten Fällen eine Gewichtsangabe je Rechnungsposition zu finden ist, kann der Wert des Gesamtgewichts als Grundlage zur Schätzung des Gewichts der einzelnen Positionen herangezogen werden.

Zu den *numerischen Daten* zählen:

- Menge (100%),
- Gewicht pro Rechnungsposition (9%),
- Rechnungspositionssumme (100%).

Unter Berücksichtigung der, aufgrund der Hilfsdaten ermittelten, Umrechnungskurse bzw. Multiplikatoren, stellen die numerischen Daten die Grundlage zur Berechnung der Eigenmasse bzw. der besonderen Maßeinheit, des Rechnungsbetrags und des statistischen Werts dar.

Redundante Daten werden nicht unbedingt benötigt, dienen jedoch der Plausibilitätsprüfung. Zu diesen Daten zählen:

- Endbetrag (100%) und

- Einzelpreis (100%).

Verwaltungsdaten

Weiters enthalten Rechnungen Daten, die zwar nicht direkt für die Erstellung von Intrastat-Meldungen benötigt werden, aber die Verwaltung der Daten erleichtern. Bei diesen Daten handelt es sich um die

- Rechnungsnummer (100%),
- Rechnungspositionsnummer (73%) und
- Artikelnummer (73%).

Die Rechnungsnummer dient zur eindeutigen Identifizierung einer Rechnung eines Ausstellers und kann die Suche eines Dokuments erleichtern oder, ebenso wie die Rechnungspositionsnummer, zur Sortierung verwendet werden. Weiters können damit etwaige doppelte Einträge erkannt werden.

Mithilfe der Artikelnummer wird ein Artikel des entsprechenden Verkäufers identifiziert. Fehlende Angaben, wie z.B. das Gewicht oder die Zolltarifnummer, können nach manueller Ergänzung mithilfe der Artikelnummer später wieder abgerufen werden.

Irrelevante Daten

Daneben finden sich in den Rechnungen weitere Daten. Vorwiegend handelt es sich dabei um Referenzdaten, wie z.B. Bestellnummer und -datum, die den Unternehmen dazu dienen den Bearbeitungsprozess zu erleichtern. Da diese für die Erstellung von Intrastat-Meldungen jedoch nicht von Bedeutung sind, wird darauf nicht näher eingegangen.

Kapitel 3

Technische Grundlagen

3.1 Begriffsdefinitionen

3.1.1 Daten, Informationen, Wissen

Die Begriffe Daten, Informationen und Wissen werden in der Literatur nicht immer eindeutig voneinander abgegrenzt und teilweise synonym verwendet. Ein etabliertes Modell, das die hierarchischen Zusammenhänge dieser Begriffe visualisiert, ist die sogenannte Wissenspyramide. Diese wurde erstmals 1989 in Ackoffs Artikel *From Data to Wisdom* [1] vorgestellt. Ursprünglich bestehend aus fünf Stufen, haben sich verschiedene Varianten der Wissenspyramide gebildet. Rowleys Untersuchung [37] ergab, dass in der aktuellen Literatur zumindest ein Konsens über die vier hierarchisch gegliederten Stufen „Daten – Informationen – Wissen – Weisheit“ besteht. In Abbildung 3.1 ist dieser typische Aufbau grafisch dargestellt.

Basierend auf Rowleys Artikel und den darin zusammengefassten Varianten bestehender Begriffsdefinitionen, werden die Begriffe, wie sie im Rahmen

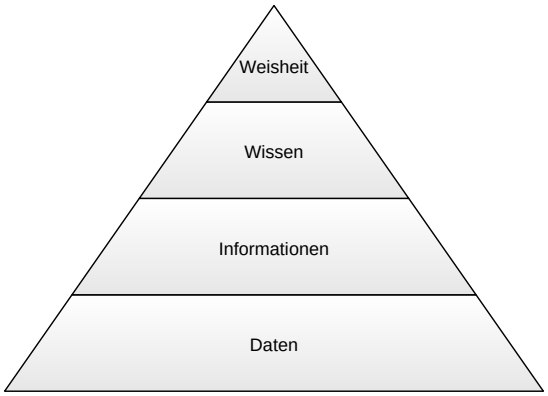


Abbildung 3.1: Wissenspyramide nach [37].

dieser Arbeit verwendet werden, nun näher definiert.

Daten werden durch Zeichen repräsentiert und sind das Resultat einer Beobachtung. Aus Daten selbst geht keine Bedeutung hervor. Um einen Nutzen zu erbringen müssen diese erst in einen Kontext gesetzt werden.

Informationen sind Daten, die durch Strukturieren und Organisieren eine Bedeutung erhalten. Informationen sind in Beschreibungen enthalten und beantworten Fragen, die mit *wer*, *was*, *wann* oder *wo* beginnen.

Wissen basiert auf Daten und Informationen, die so organisiert wurden, dass Entscheidungen getroffen werden können. Wissen ermöglicht die Beantwortung von Fragen, die mit *wie* beginnen.

3.1.2 Ontologie

Zur Erläuterung des Begriffs *Ontologie* wird häufig Grubers [21] Definition von 1993 herangezogen:

> A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose. [...] An *ontology* is an explicit specification of a conceptualization.

Dem zufolge ist eine Ontologie eine formale Beschreibung einer abstrakten und vereinfachten Sicht auf die Welt. Eine Ontologie beschreibt das Wissen eines bestimmten Fachbereichs, das für ein Wissens-basiertes System von Relevanz ist. Repräsentiert wird dieses Wissen durch ein spezielles *Vokabular*, mit dem Objekte und Beziehungen beschrieben werden.

Das World Wide Web Consortium (W3C) weist in [46] darauf hin, dass im Kontext des Semantic Web keine eindeutige Unterscheidung zwischen einer *Ontologie* und einem *Vokabular* besteht. Dennoch wird dem Begriff *Ontologie* eher eine höhere Komplexität zuerkannt:

> There is no clear division between what is referred to as „vocabularies“ and „ontologies“. The trend is to use the word „ontology“ for more complex, and possibly quite formal collection of terms, whereas „vocabulary“ is used when such strict formalism is not necessarily used or only in a very loose sense.

Ein wesentlicher Unterschied zu anderen Datenmodellen, wie z.B. relationalen oder objektorientierten, besteht darin, dass durch eine Ontologie die Semantik der modellierten Konzepte maschinell interpretierbar ist. Somit ist es einem ontologiebasierten System möglich, durch logische Schlussfolgerungen implizit enthaltene Informationen eines Datenbestands aufzufinden und diesem hinzuzufügen.

Prinzipiell wird zwischen leicht- und schwergewichtigen Ontologien (engl. lightweight and heavyweight ontologies) unterschieden. Leichtgewichtige Ontologie stellen im Wesentlichen Taxonomien dar und bestehen aus hierarchisch geordneten Klassen und Eigenschaften sowie deren Beziehungen. Durch

Axiome und Constraints sind in schwergewichtigen Ontologien zusätzlich genauere semantische Begriffsdefinition möglich [20].

3.2 Semantische Technologien

Die in diesem Kapitel vorgestellten Technologien (RDF, RDFS, OWL) werden in der Literatur üblicherweise als Technologien des *Semantic Web* bezeichnet, da diese unter der Vision desselben entwickelt werden. Die Nutzung beschränkt sich jedoch nicht ausschließlich auf Web-Dienste, weshalb in dieser Arbeit allgemein von *semantischen Technologien* gesprochen wird.

3.2.1 RDF

Das Resource Description Framework (RDF) [34] ist eine, vom W3C standardisierte, formale Sprache, die dazu dient Informationen über Ressourcen maschinenlesbar abzubilden, sodass diese Informationen ohne Bedeutungsverlust zwischen Applikationen ausgetauscht und automatisiert verarbeitet werden können. Unter einer Ressource kann in diesem Zusammenhang sowohl ein real existierendes Objekt, eine virtuelle Datei oder auch ein abstraktes Konzept verstanden werden. RDF bedient sich dabei einer Grammatik, die an natürliche Sprachen angelehnt ist und ermöglicht es Aussagen nach dem Muster „Subjekt – Prädikat – Objekt“ zu bilden. Auf diese Weise wird einer Ressource (Subjekt) eine Eigenschaft (Prädikat) mit dazugehörigen Wert (Objekt) zugewiesen. Aufgrund dieser Struktur werden Aussagen in RDF auch als *Tripel* bezeichnet.

Zur Identifizierung von Ressourcen werden Uniform Resource Identifiers (URIs) verwendet. Da RDF vor allem im Kontext des Semantic Web eingesetzt wird, kann jede Organisation, durch Verwaltung ihres eigenen Namensraums, davon ausgehen, dass ihre Ressourcen auch im Internet eindeutig identifizierbar sind. Das Objekt einer Aussage kann mithilfe einer URI auf eine weitere Ressource verweisen oder einen konkreten Datenwert, einen sogenannten *Literal*, enthalten. Da sämtliche Datenwerte als Textinformation angegeben werden, kann an einen Literal zusätzlich der Datentyp angehängt werden um festzulegen, wie der Wert zu interpretieren ist. Somit ist es möglich, dass ein Tripel auch mehr als drei Informationseinheiten enthält.

Das folgende Beispiel zeigt, unter Verwendung der Turtle-Syntax [10], ein einfaches Beispiel für ein RDF-Statement. Es wird ausgedrückt, dass einer Ressource mit der URI http://www.example.org/Invoice1 eine Eigenschaft http://www.example.org/hasInvoiceNumber mit dem Wert „2012/549“ zugewiesen ist.

```
1 @Prefix ex: <http://www.example.org/> .
2 ex:Invoice21 ex:hasInvoiceNumber "2012/549" .
```

Präfix	Namensraum
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
owl	http://www.w3.org/2002/07/owl#
xsd	http://www.w3.org/2001/XMLSchema#
ex	http://www.example.org/

Tabelle 3.1: Namensräume und Präfixe.

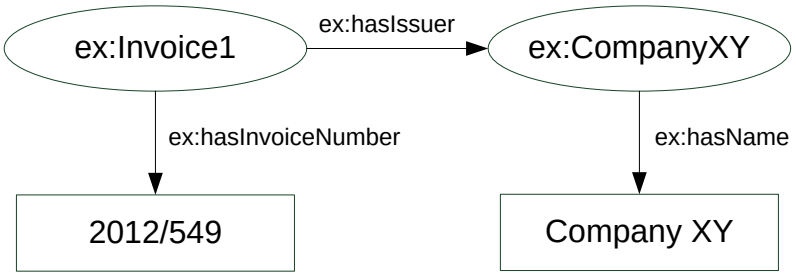


Abbildung 3.2: Beispiel für einen RDF-Graph.

Wie aus diesem Beispiel hervorgeht, erlaubt die Turtle-Syntax die Abkürzung von Namensräumen. Die Tabelle 3.1 enthält eine Übersicht der in dieser Arbeit verwendeten Namensräume sowie deren Präfix.

RDF-Graph

Die Datenstruktur in RDF entspricht einem Graph, der aus Knoten und Kanten besteht [28]. Subjekte und Objekte entsprechen den Knoten und Prädikate gerichteten Kanten. Aussagen können daher nur in die Richtung gelesen werden, in die der Pfeil zeigt. In Abbildung 3.2 ist ein einfaches Beispiel für einen RDF-Graph zu sehen. Dabei wird einer Rechnung (`ex:Invoice1`) ein Aussteller (`ex:CompanyXY`) zugewiesen. Beide Ressourcen sind wiederum mit einem Literal verbunden.

Neben URI-referenzierten Ressourcen besteht alternativ die Möglichkeit *Blank Nodes* (BNodes) zu verwenden. Dabei handelt es sich um Knoten, die nicht anhand einer URI identifiziert und als Hilfskonstrukte eingesetzt werden. Auf diese Variante wird gelegentlich zurückgegriffen, wenn die Existenz eines Knotens an einen Hauptknoten gebunden ist. Aufgrund des fehlenden Namensraums kann eine global eindeutige Identifizierung nicht sichergestellt werden und daher ist bei der Kombination verschiedener RDF-Graphen darauf zu achten, dass alle BNodes unterschiedlich identifiziert werden.

Literale

Es wird zwischen typisierten und untypisierten Literalen unterschieden. Untypisierte Literale werden im Allgemeinen für jegliche Art von Textinformationen verwendet. Eine nicht vorhandene Angabe eines Datentyps impliziert den Datentyp *String*. Um einen Hinweis auf die Sprache eines Textes zu geben, kann an einen Literal optional ein Sprachcode angehängt werden:

```
1 "Zum Beispiel"@de
```

Literale, die keinem Text ensprechen, werden typisiert. Dazu wird eine URI-Referenz an einen Literal angehängt. RDF bietet selbst keine Definition für Datentypen und empfiehlt für Standard-Datentypen die Definitionen laut XML-Schema [11] zu verwenden. Dementsprechend werden die häufigsten Datentypen Integer, Float, Date und Boolean folgendermaßen verwendet:

```
1 "129"^^xsd:integer
2 "24.99"^^xsd:float
3 "2012-08-23"^^xsd:date
4 "true"^^xsd:boolean
```

Durch die Angabe des Datentyps ist definiert, wie der Inhalt zu interpretieren ist. Es wird also zwischen der lexikalen Textform, codiert durch eine Reihe von Unicode-Zeichen, und dem dadurch ausgedrückten Wert innerhalb eines gültigen Wertebereichs unterschieden.

RDF/XML

Das W3C spezifiziert in [9] mit RDF/XML ein XML-basiertes Austauschformat für RDF-Graphen. Das folgende Codebeispiel zeigt, wie sich der, in Abbildung 3.2 dargestellte, RDF-Graph als RDF/XML repräsentieren lässt.

```
1  <?xml version="1.0"?>
2  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3           xmlns:ex="http://www.example.org/">
4
5    <rdf:Description rdf:about="http://www.example.org/Invoice1">
6      <ex:hasInvoiceNumber>2012/549</ex:InvoiceNumber>
7      <ex:hasIssuer rdf:resource="http://www.example.org/CompanyXY">
8    </rdf:Description>
9
10   <rdf:Description rdf:about="http://www.example.org/CompanyXY">
11     <ex:hasName>Company XY</ex:hasName>
12   </rdf:Description>
13 </rdf:RDF>
```

Triple Stores

Während RDF/XML ein geeignetes Format für den Informationsaustausch zwischen Applikationen bietet, ist es nicht dafür geeignet Applikationen einen

effizienten Zugang zu großen Datenmengen zu ermöglichen. Für solche Zwecke werden spezielle Datenbanksysteme eingesetzt, die als *Triple Stores* bezeichnet werden.

3.2.2 RDFS

Während RDF eine Syntax spezifiziert, bietet RDF-Schema (RDFS) [12] ergänzend dazu ein standardisiertes Grundvokabular zur Modellierung von leichtgewichtigen Ontologien bestehend aus Klassen, Eigenschaften sowie deren Beziehungen. RDFS wurde wiederum in RDF formuliert und besteht aus Klassen und Eigenschaften.

Ressourcen können in Klassen eingeteilt werden, wobei Klassen selbst wiederum Ressourcen sind. Die Mitglieder einer Klasse werden als deren Instanzen bezeichnet. Durch die Verwendung der Eigenschaft `rdf:type` als Prädikat einer Aussage wird ausgedrückt, dass eine Ressource die Instanz einer Klasse ist:

```
1 ex:Austria rdf:type ex:Country .
```

Um auszudrücken, dass es sich bei einer Ressource um eine Klasse oder eine Eigenschaft handelt, wird diese als `rdfs:Class` bzw. `rdf:Property` instanziert:

```
1 ex:Place rdf:type rdfs:Class .
2 ex:hasPopulation rdf:type rdf:Property .
```

Mithilfe der Eigenschaften `rdfs:subClassOf` bzw. `rdfs:subPropertyOf` lassen sich Klassen und Eigenschaften hierarchisch gliedern:

```
1 ex:Country rdfs:subClassOf ex:Place .
2 ex:City rdfs:subClassOf ex:Place .
```

Indem eine Instanz der Klasse `ex:City` erzeugt wird, impliziert dies, aufgrund der Klassenhierarchie, ebenso eine Zugehörigkeit zur Klasse `ex:Place`.

Die Eigenschaften `rdfs:domain` bzw. `rdfs:range` dienen weiters dazu, Beziehungen zwischen Klassen und Eigenschaften zu modellieren und den Anwendungsbereich einer Eigenschaft einzuschränken. Mit den folgenden Aussagen wird ausgedrückt, dass die Eigenschaft `ex:hasPopulation` nur einer Instanz der Klasse `ex:Place` zugewiesen werden kann und als Wert der Datentyp `xsd:integer` erwartet wird:

```
1 ex:hasPopulation rdfs:domain ex:Place .
2 ex:hasPopulation rdfs:range xsd:integer .
```

Hierbei ist darauf zu achten, dass die Klasse möglichst hoch in der Hierarchie angesiedelt ist, sodass durch Schlussfolgerungen keine Inkonsistenzen entstehen. Als Beispiel werden die folgenden Aussagen angenommen:

```
1 ex:hasPopulation rdfs:domain ex:City .
2 ex:Austria rdf:type ex:Country .
3 ex:Austria ex:hasPopulation "8000000"^^xsd:integer .
```

Die Eigenschaft `ex:hasPopulation` ist auf die Klasse `ex:City` beschränkt, wird aber dennoch für ein `ex:Country` genutzt. Aus diesen Aussagen würde

```
1 ex:Austria rdf:type ex:City .
```

folgen, da aufgrund der definierten `rdfs:domain` für jedes Subjekt einer Aussage angenommen wird, dass es sich um eine `ex:City` handelt. Analog gilt für `rdfs:range` das gleiche für das Objekt einer Aussage. Daher ist bei der Verwendung einer Ontologie darauf zu achten, dass das Vokabular entsprechend der definierten Semantik eingesetzt wird.

3.2.3 OWL

Die Web Ontology Language (OWL) [35] ist eine Sprache zur Modellierung von schwergewichtigen Ontologien und wurde, ebenso wie RDF und RDFS, durch das W3C standardisiert. OWL kann in drei verschiedenen Varianten mit steigender Komplexität und Ausdrucksstärke genutzt werden:

- OWL Lite,
- OWL DL und
- OWL Full.

OWL Lite besteht aus einem eingeschränkten Vokabular und ist nur für die Erstellung von einfachen Ontologien geeignet. Kardinalitätseinschränkungen werden zwar unterstützt, allerdings können hierfür nur die Werte 0 und 1 verwendet werden. OWL Full hingegen bietet eine maximale Ausdrucksstärke, allerdings zu Lasten maschineller Berechenbarkeit. Aufgrund der Komplexität der Sprache erachtet es das W3C als unwahrscheinlich, dass eine Reasoner sämtliche Sprachbestandteile unterstützt [35].

Aufgrund der Einschränkungen durch OWL Lite und der mangelnden Softwareunterstützung für OWL Full, stellt OWL DL einen Kompromiss zwischen Ausdrücksstärke und maschineller Berechenbarkeit dar. In diesem Kapitel werden die Bestandteile OWL DLs vorgestellt. Allgemein ermöglicht OWL eine detailliertere Spezifikation eines Wissensgebietes als dies unter Verwendung von RDFS möglich ist. Das durch OWL definierte Vokabular bezieht jedoch auch Begriffe aus RDFS mit ein. Unter anderem wurden `rdfs:subClassOf`, `rdfs:subPropertyOf`, `rdfs:domain` und `rdfs:range` übernommen.

Klassen

Alle Klassen einer OWL-basierten Ontologie werden von der gemeinsamen Superklasse `owl:Thing` abgeleitet. Es ist möglich, dass eine Ressource eine Instanz mehrerer Klassen ist. Aufgrund der semantischen Definitionen ist eine Ressource ebenso eine Instanz aller hierarchisch höher liegenden Klassen. `ex:Austria` ist beispielsweise eine Instanz der Klasse `ex:Country`. Da

`ex:Country` eine Subklasse von `ex:Place` ist, ist `ex:Austria` auch eine Instanz der Klasse `ex:Place`.

Dieses Konzept ist unter anderem aus der objektorientierten Programmierung bekannt. Im Gegensatz dazu ist es in OWL ebenso zulässig, dass eine Ressource eine Instanz mehrerer Klassen unterschiedlicher Hierarchiezweige ist. Semantisch sind die folgenden Aussagen

```
1 ex:Austria rdf:type ex:Car .
2 ex:Austria rdf:type ex:Vehicle .
```

offensichtlich falsch. Um diese Tatsache auch einer maschinellen Interpretation zugänglich zu machen, ist es erforderlich die Semantik von Begriffen näher zu definieren. Dazu werden Konstrukte aus der Mengenlehre verwendet und auf die Klassen einer Ontologie angewandt. Somit können Disjunktionen (`owl:disjointWith`), Schnittmengen (`owl:intersectionOf`), Vereinigungsmengen (`owl:unionOf`) oder Differenzmengen (`owl:complementOf`) gebildet werden.

Die Klassen `ex:Place` und `ex:Vehicle` müssen als disjunkte Mengen deklariert werden, sodass ein Reasoner die Inkonsistenz der genannten Aussagen erkennt. Inkonstistenzen treten auf, wenn Aussagen nicht den semantischen Definitionen einer Ontologie entsprechen.

```
1 <owl:Class rdf:about="#Place">
2   <rdfs:subClassOf rdf:resource="&owl;Thing"/>
3   <owl:disjointWith rdf:resource="#Vehicle"/>
4 </owl:Class>
```

Weiters können mithilfe von OWL die Bestandteile einer Klasse beschrieben werden. Durch die Angabe von Kardinalitäten wird die Anzahl der Ausprägungen einer Eigenschaft eingeschränkt. Dazu können minimale (`owl:minCardinality`), maximale (`owl:maxCardinality`) oder exakte (`owl:cardinality`) Werte festgelegt werden. Um auszudrücken, dass eine Klasse mindestens eine Adresse haben muss, wird folgendes Konstrukt der Klassendefinition beigefügt:

```
1 <owl:Restriction>
2   <owl:onProperty rdf:resource="#hasAddress"/>
3   <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">
4   1
5   </owl:minCardinality>
6 </owl:Restriction>
```

Zusätzlich kann der erwartete Datentyp einer Eigenschaft definiert werden. Dazu wird die höchte zulässige Klasse innerhalb einer Hierarchie ausgewählt. Um weiters festzulegen, dass mit der Eigenschaft `hasAddress` nur Instanzen der Klasse `Address` verbunden sein dürfen, muss die folgende Zeile dem `owl:Restriction`-Konstrukt hinzugefügt werden:

```
1 <owl:someValuesFrom rdf:resource="#Address"/>
```

Eigenschaften

Weiters ermöglicht OWL die Semantik von Eigenschaften näher zu definieren. Eigenschaften können als *transitiv*, *symmetrisch* und *funktional* deklariert werden. Mithilfe von `owl:inverseOf` kann eine Eigenschaft als inverse Eigenschaft einer anderen zugewiesen werden. Auf Basis der Aussagen

```
1 ex:hasParent owl:inverseOf ex:isParentOf .
2 ex:XYZ ex:hasParent ex:ABC .
```

kann daher die Schlussfolgerung

```
1 ex:ABC ex:isParentOf ex:XYZ
```

getroffen werden.

3.3 Regelbasierte Systeme

Regelbasierte Systeme bestehen aus einer Menge von *Fakten*, *Regeln* und einem *Interpreter*. Fakten entsprechen Aussagen, die für wahr gehalten werden, und bilden die Grundlage für weitere Schlussfolgerungen. Fakten werden üblicherweise als „Objekt-Attribut-Wert“ Tripel repräsentiert. Bezugnehmend auf das vorhergehende Kapitel, kann es sich dabei beispielsweise um einen RDF-Graphen handeln. Regeln werden nach der Form „Wenn-Dann“ gebildet

```
IF condition(s) THEN conclusion(s)
```

und definieren die zu treffenden Schlussfolgerungen, wenn bestimmte Fakten vorliegen.

Ein Interpreter wendet die vorhandenen Regeln in mehreren Zyklen auf die Fakten an. Trifft die Bedingung einer Regel zu, werden die entsprechenden Schlussfolgerungen den Fakten hinzugefügt. Sobald in einem Durchlauf keine weitere Regel mehr angewandt werden kann, wird der Vorgang beendet [25].

Bezüglich der Vorgehensweise wird zwischen *Forward-* und *Backward-Chaining* unterschieden. Beim Forward-Chaining wird, ausgehend von einigen grundlegenden Fakten, versucht, Schlussfolgerungen auf einer höheren Ebene zu treffen. Im Gegensatz dazu wird beim Backward-Chaining von einer Hypothese ausgegangen, die anhand der vorliegenden Fakten versucht wird zu verifizieren. Forward-Chaining entspricht daher einem Bottom-Up- und Backward-Chaining einem Top-Down-Ansatz [19].

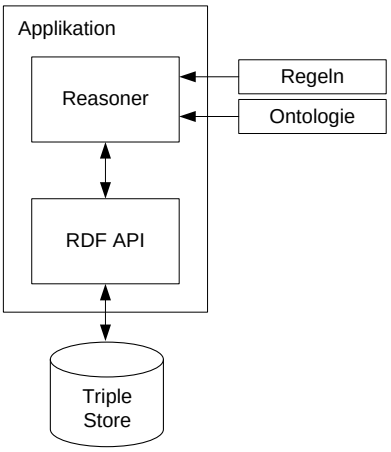


Abbildung 3.3: Aufbau RDF-basierter Systeme.

3.4 RDF-Frameworks

RDF-basierte Systeme bestehen im Wesentlichen aus folgenden Komponenten:

- Ein *Triple Store* als Datenbank zur dauerhaften Speicherung von RDF-Graphen,
- Eine *API* zur Bearbeitung von RDF-Graphen zur Laufzeit,
- Eine *Ontologie* und/oder eine Menge von *Regeln* zur formalen Repräsentation des relevanten Domänenwissens und
- Ein *Reasoner* dient der Anwendung des definierten Wissens auf einen RDF-Graphen zur Ableitung impliziter Informationen.

RDF-Frameworks, wie z.B. Jena oder Sesame, bieten Standardlösungen für die meisten dieser Bereiche. Die Implementierungen der Reasoner der RDF-Frameworks entsprechen jedoch nicht immer den Anforderungen der Entwickler. Gründe dafür können die mangelnde Unterstützung bestimmter Regelsprachen, geringe Performance oder unvollständiges Reasoning sein. Daher können spezielle Reasoner, die wiederum auf den APIs aufbauen, eingesetzt werden.

3.5 Dokumentenverarbeitung

Systeme zur automatisierten Verarbeitung von Dokumenten werden in der Literatur auch als *Document (Analysis and) Understanding* Systeme bezeichnet. Dengel definiert in [16] *Document Understanding* als ein Gebiet, das sich mit der logischen und semantischen Analyse von Dokumenten beschäftigt. Ziel ist es dabei aus Dokumenten Informationen so zu extrahieren, dass diese

für Menschen verständlich, sowie für Maschinen interpretierbar sind. Durch die Anwendung von Wissen auf die, in Form einer Rastergrafik vorliegenden, Daten werden Informationen aus Dokumenten extrahiert. Folgende Wissensbereiche, auf die im Zuge der Extraktion zurückgegriffen werden kann, wurden identifiziert:

- Vokabular einer Sprache,
- Layout einer Dokumentenklasse,
- Art der Nachricht,
- Erwartete Informationen,
- Informationen aus verfügbaren Datenbanken.

Prinzipiell kann die Analyse von Dokumenten auf Layout- und Textebene erfolgen.

Layout

Hinsichtlich des Layouts können Dokumente als *statisch*, *strukturiert* oder *unstrukturiert* klassifiziert werden.

Statische Dokumente entsprechen Formularen mit fixem Layout, sodass jedes Dokument exakt gleich aufgebaut ist. Der bekannte Aufbau ermöglicht eine gezielte Extraktion von Informationen innerhalb eines vordefinierten Bereichs.

Strukturierte Dokumente bestehen aus Informationsblöcken mit variablen Längen und können ebenso optionale Bestandteile aufweisen. Im Gegensatz zu statischen Dokumenten kann die absolute Position einzelner Informationsblöcke von Dokument zu Dokument deutlich voneinander abweichen. Dennoch besteht eine erkennbare Struktur zwischen Informationsblöcken, Schlüsselwörtern und Datenfeldern, sodass Informationen gezielt extrahiert werden können.

Unstrukturierte Dokumente weisen keine bzw. wenig Merkmale auf, die bei der Suche nach Informationen genutzt werden können. Sie bestehen weitgehend aus Fließtext, sodass keine offensichtliche Zuordnung zwischen Schlüsselwörtern und Datenfeldern besteht. Um Informationen aus diesen Dokumenten zu extrahieren, ist es notwendig den enthalten Text semantisch zu analysieren.

In [27] wird auf den Unterschied zwischen physischer und logischer Struktur eines Dokuments hingewiesen. Die physische Struktur entspricht der Einteilung einer Seite in zusammenhängende Blöcke, die aus Zeilen, die wiederum aus Wörtern bestehen. Die logische Struktur hingegen gliedert ein Seite in logische Einheiten wie Überschriften, Tabellen oder Listen.

Kapitel 4

Verwandte Arbeiten

In diesem Kapitel werden einige thematisch verwandte Arbeiten vorgestellt, die sich ebenfalls mit der Informationsextraktion aus Rechnungen befassen und somit einige Überschneidungspunkte zu dieser Arbeit bieten.

Kategorisierung

Generell können Dokumentenverarbeitungssysteme für eine Reihe unterschiedlicher Szenarien zum Einsatz kommen. Der kleinste gemeinsame Nenner besteht dabei in der Informationsextraktion aus digitalisierten Dokumenten. Da es sich um ein breit gefasstes Gebiet handelt, das sowohl die Verarbeitung historischer Dokumente [36] als auch technischer Skizzen [41] umfasst, wird der Begriff *Dokument* auf solche beschränkt, die größtenteils aus, in Standardschrift gedrucktem, Text bestehen und somit keine speziellen Anforderungen bezüglich Texterkennung oder Bildverarbeitung stellen. Dokumentenverarbeitungssysteme können anhand folgender Merkmale kategorisiert werden:

- *Anwendungsbereich*: Hinsichtlich des Anwendungsbereichs wird zwischen Single-Domain- und Multi-Domain-Systemen unterschieden. Während sich Single-Domain-Systeme auf ein Anwendungsgebiet, wie z.B. die Verarbeitung von Rechnungen beschränken, sind Multi-Domain-Systeme in der Lage mit verschiedenartigen Dokumenten, z.B. Geschäftsdokumenten im Allgemeinen, umzugehen.
- *Dokumentenart*: Ein weiteres Unterscheidungsmerkmal betrifft die unterstützte Dokumentenart. Es wird zwischen Formularen, strukturierten und unstrukturierten Dokumenten unterschieden.
- *Lernende bzw. Nicht-lernende Systeme*: Weiters kann zwischen lernenden und nicht-lernenden Systemen unterschieden werden. Prinzipiell können Lernalgorithmen für verschiedene Vorgänge im System eingesetzt werden. In diesem Kontext wird ein System als lernend bezeichnet, wenn es fähig ist Informationen aus unbekannten Dokumenten zu extrahieren.

- *Automatische bzw. halbautomatische Systeme*: Während automatische Systeme weitgehend selbständig agieren, erfordern halbautomatische Systeme an wesentlichen Stellen des Systemablaufs Interaktionen des Benutzers.
- *Dokumentenklassifizierung*: Es gibt eine Reihe unterschiedlicher Methoden Dokumente zu klassifizieren. Systeme unterscheiden sich durch die Merkmale, die bei der Klassifizierung berücksichtigt werden, deren Repräsentation und dem Klassifizierungsalgorithmus. Chen et al. bieten in [14] einen umfassenden Überblick zum Thema Dokumentenklassifizierung.

4.1 smartFIX

Smart for Information Extraction (smartFIX) ist ein kommerzielles Dokumentenverarbeitungssystem, das von Insiders Technologies[1] entwickelt wird und ursprünglich am Deutschen Forschungszentrum für Künstliche Intelligenz (DKFI) entstanden ist. In [26], einer Publikation aus dem Jahr 2004, wird der Systemaufbau vorgestellt. Zu diesem Zeitpunkt wurde mit smartFIX healthcare der Fokus auf die Verarbeitung von medizinischen Rechnungen für Versicherungsunternehmen gelegt.

Mithilfe eines grafischen Editors, dem *DocumentManager*, werden Dokumentenklassen erstellt. Anhand eines Beispielbildes einer Rechnungen können Regionen definiert werden, innerhalb dieser nach bestimmten Informationen gesucht wird. Einer Region wird ein gesuchtes logisches Objekt, z.B. der Rechnungsbetrag, zugewiesen. Weiters wird definiert, wie die gesuchten Daten innerhalb der Region strukturiert sind, um diese zu extrahieren. Neben vordefinierten Datentypen wie *Address*, *Check Box* oder *Barcode* können manuell Key-Value-Beziehungen, Tabellenstrukturen oder reguläre Ausdrücke definiert werden. Die Verarbeitung von Dokumenten gliedert sich in folgende Schritte:

1. Image Preprocessing,
2. Klassifizierung des Dokuments,
3. Informationsextraktion,
4. Nachbearbeitung,
5. Validierung.

Um dem Benutzer einen Hinweis auf die Zuverlässigkeit der Daten zu bieten, wurde eine dreistufige Bewertungsskala eingeführt. Diese besteht aus den Stufen *safe*, *proposal* und *OCR result*.

[1] http://www.insiders-technologies.de

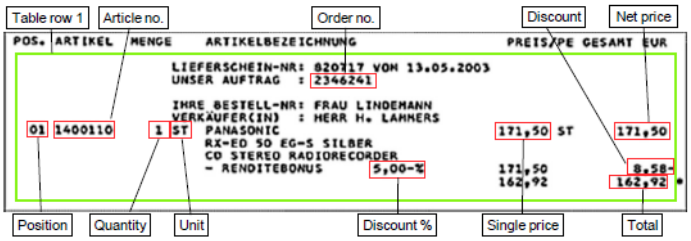


Abbildung 4.1: Tabelle mit unklarer Struktur (übernommen aus [15]).

Komplexe Tabellen

In einem aktuellen Paper aus dem Jahr 2011 [15] beschäftigen sich Deckert et al. mit der Informationsextraktion aus komplexen Tabellen. Wie aus Abbildung 4.1 hervorgeht, können Rechnungen in der Praxis unklar strukturierte Tabellen enthalten, wodurch die Extraktion der benötigten Daten erschwert wird.

Infolge der Klassifizierung wird ein Dokument einem Geschäftsprozess zugeordnet. Mit einem Geschäftsprozess ist ein Datenmodell verbunden, das die Erwartungen, an die in einem Dokument enthaltenen, Informationen definiert. Im Falle einer Rechnung ist es dem System somit erkennbar, dass z.B. eine Tabelle, bestehend aus Artikelnummer, Menge, Einzel- und Gesamtpreis erwartet wird. Für jede Spalte einer Tabelle kann der erwartete Datentyp definiert werden. Weiters können Erwartungen an das Layout gestellt werden. Alle Spaltenüberschriften sollen z.B. eine ähnliche vertikale Position aufweisen und die Einträge einer Spalte linksbündig geordnet sein.

Auf Basis der Erwartungen werden für jede Spalte potentielle Kandidaten ermittelt. Ein Qualitätsmaß beschreibt wie gut die Kriterien erfüllt werden und ermöglicht es die Menge der Kandidaten weiter zu reduzieren.

Es wird darauf hingewiesen, dass aufgrund der vielfältigen Unterschiede im Layout, die definierten Erwartungen nicht für sämtliche Rechnungen gültig sein können.

Der Algorithmus wurde anhand von 3366 Rechnungen, bestehend aus 19190 Zellen, evaluiert. Dabei wurde eine Erkennungsrate von 68,9% erzielt. Davon konnten 59,9% der extrahierten Daten verifiziert werden und bedurften keiner weiteren Überprüfung durch einen Benutzer. 0,85% wurden jedoch fälschlicherweise als korrekt bewertet.

4.2 Multi-Class Invoices

Cesarini et al. entwickelten ein Rechnungsverarbeitungssystem [13], bei dem das Wissen zur Informationsextrakion in zwei Ebenen eingeteilt wird. Im Klassenunabhängigen Wissen (engl. *Class Independent Domain Knowledge*,

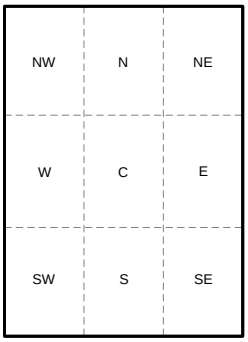


Abbildung 4.2: Einteilung eines Dokuments in neun Sektoren (nach [13]).

CIDK) werden die Charakteristiken von logischen Objekten zusammengefasst. Dieses Wissen kommt bei der Verarbeitung unbekannter Dokumente zur Anwendung. Im Klassenabhängigen Wissen (engl. *Class Dependent Domain Knowledge*, CDDK) werden hingegen die Charakteristiken einer besimmten Dokumentenklasse abgebildet. CIDK und CDDK werden anhand einer Menge von Beispielrechnungen konstruiert und bestehen aus den Beschreibungen der enthaltenen logischen Objekte. Diese werden durch folgende Eigenschaften repräsentiert:

- Name,
- Datentyp,
- Mögliche Schlüsselwörter,
- Absolute Positionen der Schlüsselwörter und Datenfelder,
- Relative Positionen der Datenfelder zu Schlüsselwörtern,
- Durchschnittliche Größe des Datenfelds.

Zur Beschreibung der absoluten Positionen wird eine Seite in neue gleichmäßige Bereiche eingeteilt, die entsprechend den Himmelsrichtungen (NW, N, NE,...) benannt sind. Eine grafische Darstellung dazu findet sich in Abbildung 4.2. Anhand des Mittelpunkts einer Region erfolgt die Zuordnung zu einem dieser Sektoren. Im CIDK wird die Häufigkeit, mit der ein Objekt in einem der neun Sektoren gefunden wurde, gespeichert.

Die relative Position zu einer weiteren Region wird durch die Beziehungen *above*, *below*, *rightOf* bzw. *leftOf* beschrieben. Ausgehend vom Mittelpunkt der Referenzregion wird das Dokument, wie aus Abbildung 4.3 hervorgeht, in vier Bereiche eingeteilt.

Systemablauf

Der Systemablauf erfolgt in drei Phasen:

- *Document Analysis*: Zunächst werden Text- und Layout-Informationen

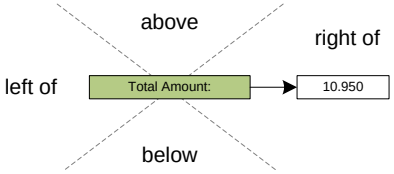


Abbildung 4.3: Relative Positionen (nach [13]).

aus der Rechnung extrahiert. Das Ergebnis dieser Phase ist eine Liste von Wörtern und deren Koordinaten. Token werden als numerisch, alphanumerisch oder alphabetisch klassifiziert.

- *Document Classification*: Auf Basis der Layout-Informationen wird versucht das Dokument einer bekannten Klasse zuzuordnen. Der dafür verwendete Algorithmus wird ausführlich in [2] beschrieben.
- *Document Understanding*: In dieser Phase wird versucht die logischen Objekte anhand der Beschreibungen zu finden und zu extrahieren. Sofern das Dokument klassifiziert werden konnte, kommt neben dem CIDK auch das entsprechende CDDK zur Anwendung. Zuerst wird versucht ein Datenfeld auf Basis des Schlüsselwortes und der relativen Position zu diesem zu lokalisieren. Gelingt dies nicht, da z.B. das Schlüsselwort nicht erkannt wurde, wird das Datenfeld laut der Beschreibung der absoluten Position innerhalb des CDDK bzw. CIDK gesucht.

Ergebnisse

Evaluiert wurde das System wurde auf Basis von insgesamt 244 Rechnungen von 83 unterschiedlichen Rechnungsausstellern. Die Rechnungen wurden mit 300 dpi als Schwarz-Weiß-Bild gescannt. Extrahiert wurden jeweils die Rechnungsnummer und der Endbetrag. Es wurden verschiedene Tests durchgeführt, wobei vergleichbare Resultate erzielt wurden. Rund 80% der Rechnungsnummern und 90% der Endbeträge wurden korrekt extrahiert. Als häufigste Fehlerursachen wurden Segmentierungs- und OCR-Fehler genannt.

4.3 CBRDIA

Hamza et al. [22] verfolgen den Ansatz des Fall-basierten Schließens (engl. *Case-based Reasoning*, CBR) in ihrem System „Case-based Reasoning for Document Invoice Analysis“ (CBRDIA). CBR ist eine Problemlösungsstrategie, die es ermöglicht bekannte Problemlösungen für neue, unbekannte Probleme zu nutzen. Dementsprechend werden in CBRDIA Dokumente ebenfalls in bekannte und unbekannte Klassen eingeteilt. Dokumente einer Klasse weisen dieselbe Struktur auf, sodass die relative Anordnung von Informations-

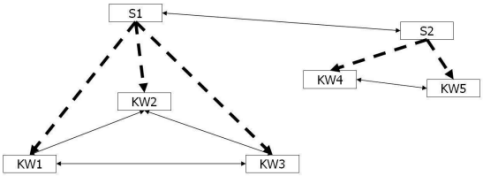


Abbildung 4.4: Beispiel für einen einfachen Dokumentengraph in CBRDIA (übernommen aus [22])

blöcken übereinstimmt. Folgende Phasen treten bei der Verarbeitung eines Dokuments auf:

- Problem Elaboration,
- Global Solving,
- Local Solving.

Problem Elaboration

Zunächst liegt ein Dokument als eine Menge von Wörtern und deren Koordinaten vor. Um in weiterer Folge entscheiden zu können, ob das Dokument einer bekannten Klasse zuordenbar ist, wird im ersten Schritt eine Problembeschreibung formuliert. Dazu wird das Dokument schrittweise in einen Graph, bestehend aus *Keyword Structures* (KWS) und *Pattern Structures* (PS) überführt. KWS sind Informationsblöcke bestehend aus einer Anordnung von Schlüsselwörtern (wie z.B. „Gesamtbetrag“) und dazugehörigen Datenfeldern, deren Positionen zunächst noch nicht bekannt ist. PS werden verwendet um Tabellen mit beliebig vielen Einträgen abzubilden. Dazu wird die Struktur der enthaltenen Daten analysiert. Eine Tabelle kann beispielsweise als $ABAAAA$ codiert werden. Damit wird der Datentyp der einzelnen Felder einer Tabelle entweder als Text (A) oder als Zahl (B) definiert. Abbildung 4.4 enthält ein einfaches Beispiel für einen Dokumentengraph in CBRDIA. Dieser definiert die Anordnung zweier KWS und den relativen Positionen der darin enthaltenen Schlüsselwörter (KW1 bis KW5).

Global Solving

Auf Basis des Dokumentengraphs wird in der Datenbank nach einem bekannten Fall, repräsentiert durch einen gleichen Graphen, gesucht. Wird eine Lösung gefunden, können die im Dokument enthaltenen Informationen anhand der Lösungsbeschreibung ausgelesen werden. Diese enthält für jedes Schlüsselwort eine relative Positionsangabe und den erwarteten Datentyp des dazugehörigen Datenfelds.

Local Solving

Handelt es sich um ein unbekanntes Dokument und somit um eine neues Problem, wird versucht mittels *Local Solving* eine Lösung zu finden. Dazu wird das Dokument in mehrere Teilprobleme eingeteilt. Für jedes Teilproblem wird wiederum, auf Basis der bekannten Strukturen innerhalb der bekannten Dokumentenklassen, nach einer Lösung gesucht.

Ergebnisse

Global Solving wurde anhand von 100 Dokumenten evaluiert, wobei 85% der Informationen korrekt extrahiert wurden. 11% wurden aufgrund fehlerhafter Texterkennung nicht korrekt erkannt und bei den verbleibenden 4% handelt es sich um sogenannte Systemfehler. Darunter werden falsche, nicht vorhandene oder verwechselte Informationen zusammengefasst.

Local Solving wurde anhand einer Menge von 923 Dokumenten evaluiert, wobei rund 75% der Informationen korrekt extrahiert wurden. 13% wurden aufgrund von OCR-Fehler nicht erkannt und weitere 12% entfielen auf Systemfehler.

Wie aus den Ergebnissen hervorgeht, eignet sich das System prinzipiell um bekannte und unbekannte Dokumente zu verarbeiten. Da keine Implementierungsdetails bzw. keine konkreten Systemkomponenten genannt werden, bleibt offen, inwiefern die Texterkennungsfehler reduziert werden können. Weiters sind die in der Arbeit abgebildeten Dokumente einfach strukturiert. Rechnungspositionen sind durchgehend einzeilig und Tabellen sind in eindeutig voneinander abgegrenzte Spalten eingeteilt. Inwieweit komplexer strukturierte bzw. mehrseitige Dokumente unterstützt werden, geht auch nicht näher hervor.

4.4 Repeated Structure

Bart et al. [8] entwickelten einen Algorithmus um Informationen, anhand von sich wiederholenden Mustern, aus Dokumenten zu extrahieren. Das System arbeitet halbautomatisch. Der Benutzer markiert zuerst manuell einen Tabelleneintrag und grenzt die darin enthaltenen Felder voneinander ab. Auf diese Weise wird ein Referenzdatensatz erstellt und im Dokument nach weiteren Einträgen mit dem gleichen Muster gesucht. Hiefür wurde ein Wahrscheinlichkeitsmodell entwickelt. Dieses basiert auf der Annahme, dass Dokumente so gestaltet sind, dass sie für den Menschen leicht verständlich sind.

Nach erfolgter Texterkennung werden die zusammenhängenden Zeichenketten, genannt *Token*, zunächst den Datenfeldern des Referenzeintrags als potentielle Kandidaten zugeordnet. Dabei werden Merkmale der Referenzdaten mit denen der Kandidaten verglichen und somit die Wahrscheinlichkeit, mit der ein Token einem Datenfeld zuzuordnen ist, ermittelt. Anhand ver-

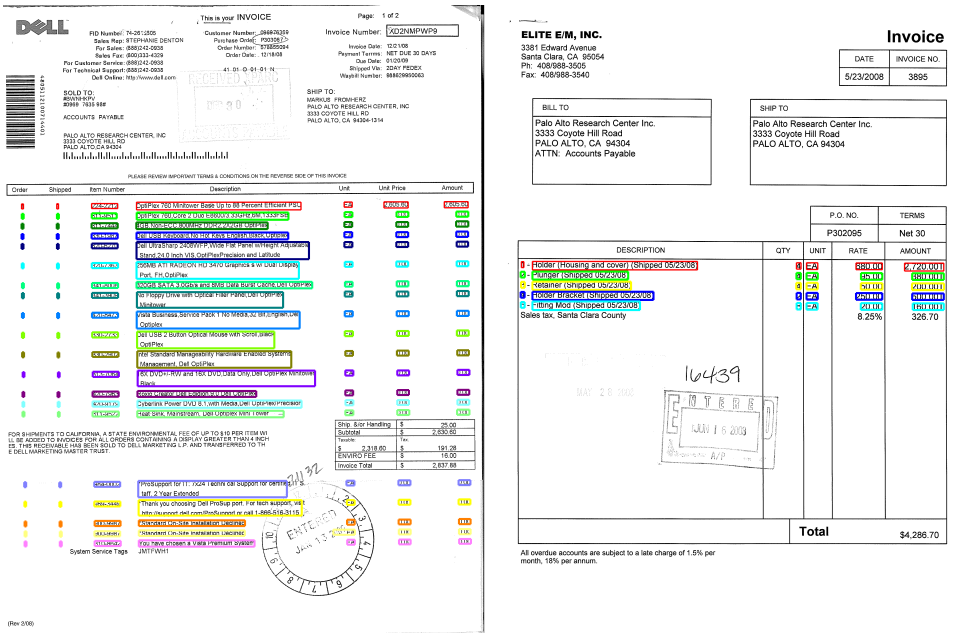


Abbildung 4.5: Beispielrechnungen aus [8]. Der erste Tabelleneintrag wurde dem System jeweils als Muster übergeben – alle weiteren automatisch erkannt.

schiedener Faktoren wird die optische Ähnlichkeit berechnet. Dazu zählen Höhe und Breite eines Tokens, die Lage im Dokument und ein Faktor, der anhand der benachbarten Token aussagt, wie hervorstechend ein Token ist. Weiters werden die üblichen Abstände zwischen den Token eines Datenfelds, sowie die Beziehungen zwischen Datenfeldern analysiert.

Um das System zu testen wurden 51 Rechnungen 15 unterschiedlicher Rechnungsaussteller verarbeitet. Zwei der verwendeten Exemplare sind in Abbildung 4.5 dargestellt. Von den 409 Rechnungspositionen wurden 92 % gefunden. Informationen aus 63 % der Rechnungen wurden vollständig extrahiert. Die restlichen 37 % enthielten meist ein bis zwei Fehler. Am häufigsten wurden Segmentierungsfehler festgestellt, wobei benachbarte Token einem Datenfeld fälschlicherweise zugeordnet wurden. Eine Fehlerquelle stellte die OCR-Komponente dar. Nicht erkannte Textregionen führten dazu, dass ein Tabelleneintrag nicht erkannt wurde, da keine Übereinstimmung mit dem Muster des Referenzeintrags mehr gegeben ist.

Zusammengefasst eignet sich dieser Algorithmus für klar und einheitlich strukturierte Dokumente. Wie aus den Beispielrechnungen hervorgeht, ist eine eindeutige Beziehung zwischen den Tabellenüberschriften und den zugehörigen Datenfeldern gegeben und es besteht eine erkennbare Abgrenzung der Spalten. Da die gesuchten Tabelleneinträge weitgehend mit dem gewähl-

ten Muster übereinstimmen müssen, können allerdings Dokumente mit unregelmäßigem Layout oder optionalen Datenfeldern nur begrenzt verarbeitet werden.

Kapitel 5

Systemarchitektur

5.1 Anforderungen

Mithilfe des Systems soll die Erstellung von Intrastat-Meldungen weitgehend automatisiert werden. Das System muss daher grundlegend in der Lage sein ein- und mehrseitige Rechnungen zu verarbeiten und die wesentlichen, zur Erstellung von Intrastat-Meldungen notwendigen, Informationen zu extrahieren. Dazu wird eine Menge von ausstellerspezifischen Extraktionsregeln verwendet, die manuell konstruiert werden.

Für den Benutzer soll der Arbeitsaufwand wesentlich reduziert werden. Daher ist es erforderlich, die extrahierten Daten automatisch auf ihre Gültigkeit hin zu überprüfen und entsprechend zu visualisieren. Der Benutzer muss die Qualität der Daten schnell einschätzen können und soll nur noch fehlende Daten ergänzen bzw. fehlerhafte Daten korrigieren. Zur Überprüfung der Daten ist es erforderlich, das entsprechende Originaldokument darzustellen. Um die Dateneingabe zunehmend zu reduzieren, soll das System manuell vorgenommene Klassifizierungen in Zukunft berücksichtigen und automatisch ergänzen.

5.2 Geplante Einschränkungen

DIN A4

DIN A4 ist ein international anerkanntes Papierformat mit einem Maß von 210 x 297 mm. Sämtliche handelsüblichen Drucker und Scanner unterstützen dieses Format, sodass Geschäftsdokumente in der Regel im DIN A4-Format vorliegen. Für Systeme, die darauf abzielen die unternehmensinterne Buchhaltung zu automatisieren, sind darüber hinaus weitere Formate, vor allem die der üblichen Kassabons, von Relevanz. Da, im Hinblick auf die Erstellung von Intrastat-Meldungen, davon auszugehen ist, dass alle relevanten Dokumente im DIN-A4-Format vorliegen und mit einem entsprechenden

Scanner gescannt werden, wurde das System auf dieses Format beschränkt. Der Vorteil liegt darin, dass, aufgrund des zu erwartenden Formats, Pixel in Zentimeter umgerechnet werden können. Das Verfassen von Regeln, bei denen Abstände zwischen Textbestandteilen eine Rolle spielen, erfolgt somit wesentlich intuitiver, als bei Verwendung von relativen Abständen in Prozentwerten des Seitenverhältnisses.

Hierbei sei jedoch angemerkt, dass eine Scanauflösung von 300 dpi zwar empfohlen, aber nicht festgeschrieben ist. Eine konkrete Anzahl von Pixeln kann daher nicht erwartet werden, weshalb das Format aufgrund des Seitenverhältnisses überprüft wird. Sämtliche DIN-Formate weisen allerdings das gleiche Seitenverhältnis auf. Somit würde das System ein gescanntes DIN-A3-Dokument ebenfalls als DIN-A4-Seite interpretieren.

Vorsortierung

Bei der Verarbeitung von mehrseitigen Dokumenten erwartet das System eine korrekte Sortierung der einzelnen Seiten. Diese Einschränkung wurde getroffen, da davon ausgegangen wird, dass Geschäftsdokumente in Büros geordnet aufbewahrt werden. Sofern Rechnungspositionen nicht auf aufeinanderfolgende Seiten verteilt sind, ist es aber ohnehin nur entscheidend, dass sich die erste und die letzte Seite eines Dokuments jeweils an der korrekten Position befindet. Ein chaotische Sortierung der dazwischenliegenden Seiten stellt unter diesen Voraussetzungen kein Problem dar.

Rechnungen

Da Intrastat-Meldungen zum größten Teil auf Basis von Rechnungen erstellt werden, wurde das System vorerst auf diese Dokumentenart beschränkt. Sofern Gutschriften eines Ausstellers das gleiche Layout, wie deren Rechnungen aufweisen, ist eine einfache Integration möglich. Der wesentliche Unterschied zu einer Rechnung besteht in diesen Fällen darin, dass unterschiedliche Schlüsselwörter verwendet werden.

5.3 Überblick

Die Kernkomponenten, auf denen das in dieser Arbeit vorgestellte System basiert, sind die OCR-Library *Tesseract*[1] und das RDF-Framework *Jena*[2]. Tesseract wurde ursprünglich von Hewlett-Packard (HP) entwickelt und im Jahr 2005 als Open Source veröffentlicht [39]. Tesseract wird in diversen Internetforen, z.B. auf stackoverflow.com, häufig als beste frei verfügbare OCR-Engine empfohlen. Kommerzielle Alternativen wurden aus Kostengründen vorläufig nicht näher in Betracht gezogen. Dennoch berichten verschiedene

[1] http://code.google.com/p/tesseract-ocr/
[2] http://jena.apache.org/

Quellen von zuverlässigeren Resultaten bei Verwendung der OCR-Engine von Abbyy[3].

Kompatibilität zwischen 32- und 64-Bit-Systemen

Tesseract steht als 32-Bit-API unter C++ zur Verfügung, während Jena hingegen auf Java basiert. Prinzipiell ist es möglich C++-Libraries über Java Native Interface (JNI) oder Java Native Access (JNA) in Java-Anwendungen zu integrieren. Da das System auf absehbare Zeit nur auf Windows-Systemen zum Einsatz kommen wird, wäre die dadurch verlorene Plattformunabhängigkeit vernachlässigbar. Nach einer Java-Integration ist jedoch ein 32-Bit-basiertes Java Runtime Environment (JRE) erforderlich, um Tesseract Java-seitig verwenden zu können. Unter Windows bedeutet dies wiederum eine Beschränkung des Arbeitsspeichers auf 1,5 GB [18]. Während der Entwicklung wurde festgestellt, dass Reasoning ein speicherintensiver Vorgang ist. Nachdem 64-Bit-basierte Systeme mittlerweile Standard sind, würde die Verwendung eines 32-Bit JRE eine unnötige Beschränkung des Systems darstellen. Daher wurde das System in zwei Anwendungen geteilt. Eine C++-Anwendung dient zur Konvertierung von gescannten Dokumenten in ein maschinenlesbares XML-Format. Diese XML-Dateien werden in der Java-basierten Hauptanwendung eingelesen und zu Intrastat-Meldungen weiterverarbeitet. Die Zweiteilung des Systems führt vorerst zu einem Komfortverlust, da ein manueller Zwischenschritt entstanden ist. In weiterer Folge ist es denkbar einen Kommunikationskanal zwischen den beiden Anwendungen zu errichten, sodass die Hauptanwendung via Netzwerkinterface über neue Dokumente informiert wird.

5.3.1 Arbeitsschritte

Bevor auf die technischen Details des Systems eingegangen wird, folgt eine Übersicht der notwendigen Arbeitsschritte aus Sicht des Anwenders. In Abbildung 5.1 sind die Vorgänge, die eine manuelle Interaktion erfordern, grafisch dargestellt.

Zunächst werden die Rechnungen gescannt und mittels Scanner-Software als PNG oder TIFF gespeichert. Diese Dateien werden von der C++-Anwendung eingelesen und nach erfolgter Texterkennung als XML gespeichert. Die XML-Dateien werden von der Hauptanwendung eingelesen und nach erfolgter Fehlerkorrektur und Schlüsselworterkennung in einen RDF-Graphen konvertiert. Auf Basis eine Menge von Regeln werden die wesentlichen Rechnungsdaten extrahiert und in vorläufige Intrastat-Daten konvertiert. Über die Benutzeroberfläche können die extrahierten Daten bearbeitet oder ergänzt werden. Anschließend werden die Intrastat-Daten in Form einer Excel-Datei exportiert. Mithilfe der Software IDEP werden diese Daten schließlich

[3] http://www.abbyy.de/

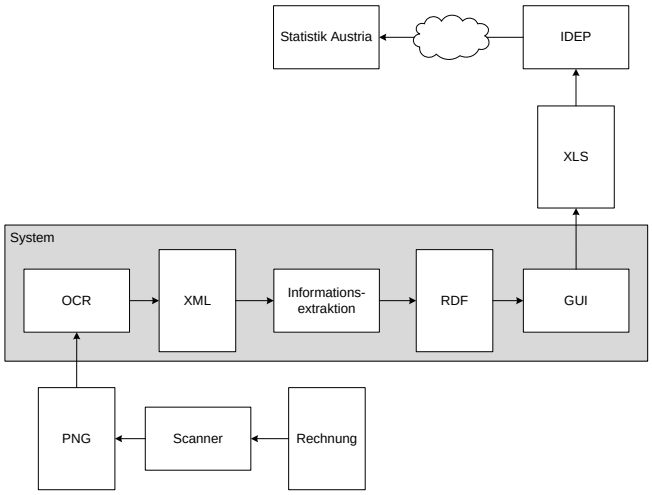


Abbildung 5.1: Übersicht der Arbeitsschritte.

als Intrastat-Meldung an die Statistik Austria gesendet.

5.3.2 C++-Anwendung

Die Anwendung zur Konvertierung von Bild- in XML-Dateien wurde in C++ geschrieben und basiert hauptsächlich auf den Open-Source-Libraries Tesseract 3.01 und OpenCV 2.3.0[4]. Tesseract wird zur Texterkennung und Layout-Analyse eingesetzt. Da Tesseract standardmäßig nur Bilder im TIFF-Format unterstützt, wird via OpenCV die Kompatibilität zu weiteren Formaten, wie PNG oder JPEG, hergestellt. Weiters wurde OpenCV zur Durchführung diverser Bildoperationen integriert. Zur Implementierung eines GUIs wurde Qt 4.7.4[5] verwendet.

5.3.3 Java-Anwendung

Das Hauptsystem zur Informationsextraktion wurde in Java entwickelt. Das System kann in mehrere aufeinander aufbauende Schichten eingeteilt werden. Ein Überblick der wesentlichen Systembestandteile wird in Abbildung 5.2 gegeben. Innerhalb einer Schicht sind bedeutende Java-Klassen in weißer bzw. APIs in blauer Farbe dargestellt.

Dateisystem

Im Dateisystem werden folgende Daten verwaltet:

[4] http://opencv.org/
[5] http://qt-project.org/

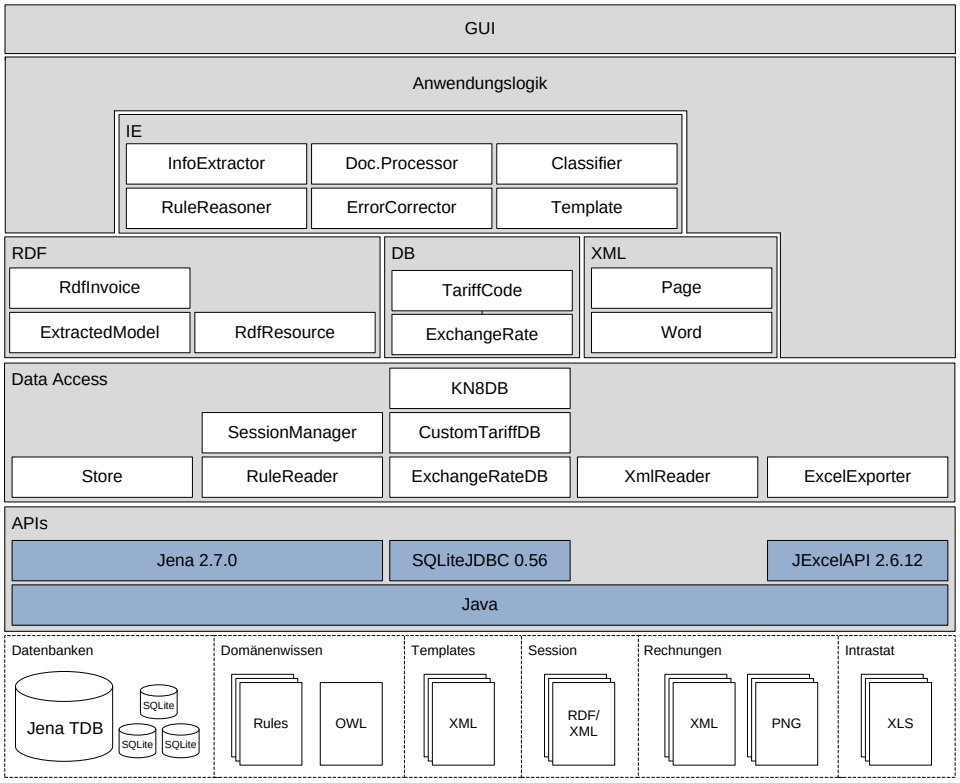


Abbildung 5.2: Systemaufbau der Java-Anwendung.

- RDF-Statements, die zur Ableitung und Verifizierung von Intrastat-Daten benötigt werden, befinden sich in einem Jena-TDB Triple-Store. Drei weitere SQLite-Datenbanken beinhalten Daten über den KN8-Warenkatalog, benutzerdefinierte Warenklassifikationen und Wechselkurse.
- Das Domänenwissen wird durch eine Ontologie und eine Menge von Regeln repräsentiert. Die Ontologie definiert auf formale Weise das Vokabular, das innerhalb des Systems zur Anwendung kommt. Regeln werden je Dokumentenklasse in getrennten Verzeichnissen verwaltet.
- Templates werden zur Dokumentenklassifizierung benötigt. Ein Template entspricht im Wesentlichen einer in XML konvertierten Seite eines Dokumentenausstellers, die um ein paar Informationen ergänzt wurden.
- Um den aktuellen Zustand der Anwendung zu speichern, werden die verarbeiteten Dokumente als RDF/XML gespeichert.
- Rechnungen, die zur Weiterverarbeitung zu Intrastat-Meldungen bestimmt sind, werden seitenweise als XML-Dateien, die je auf eine zugehörige Bilddatei verweisen, zwischengespeichert und sind das Produkt

der C++-Anwendung.
- Die exportierten Intrastat-Meldungen werden als Excel-Dateien gespeichert.

Schichtmodell

Das System basiert auf Java und verwendet unter anderem die APIs Jena 2.7.0, SQLiteJDBC 0.56[6] und JExcelAPI 2.6.12[7].

Der Data-Access-Layer dient dazu, die im Dateisystem verwalteten Daten in die Laufzeitumgebung zu integrieren und verwendet dazu Methoden der erwähnten APIs. Darauf baut eine weitere Abstraktionsschicht auf, die den Zugriff auf logische Datenobjekte ermöglicht. Diese können, entsprechend ihres Ursprungs, in die drei Gruppen *RDF*, *DB* und *XML* eingeteilt werden. Zur Informationsextraktion (*IE*) werden diese Daten miteinander kombiniert. Die IE-Schicht beinhaltet Methoden zur Dokumentenklassifizierung, OCR-Fehlerkorrektur, Schlüsselwortsuche und Informationsextraktion. Die Anwendungslogik bedient sich der zugrundeliegenden Schichten, um einzelne Anwendungsfunktionen zu implementieren. Über ein GUI werden diese schließlich dem Benutzer zugänglich gemacht.

5.4 Aufbau des Triple Stores

Folgende Schritte wurden zum Aufbau des Triple Stores durchgeführt:

- Suchen relevanter Datenquellen,
- Extraktion und Konvertierung in lokales relationales Datenschema,
- Manuelle Korrektur und Ergänzung von Daten,
- Konvertierung in lokales RDF-Modell,
- Ableitung impliziter Informationen mittels OWL-Reasoning.

Als Datenquelle wurde zuerst der SPARQL-Endpoint der DBPedia[8] herangezogen. Daten über Länder und Währungen wurden daraus extrahiert. Da DBPedia allerdings nicht zwischen historischen und aktuell existierenden Ländern unterscheidet, wurden diese Daten mit denen des CIA-Factbook verknüpft. Letztere wurden über den, mittlerweile nicht mehr erreichbaren, Endpoint[9] der Freien Universität Berlin abgerufen.

[6] http://code.google.com/p/sqlite-jdbc/
[7] http://jexcelapi.sourceforge.net/
[8] http://dbpedia.org/sparql
[9] http://www4.wiwiss.fu-berlin.de/factbook/sparql

5.5 SQLite-Datenbanken

Euro-Wechselkurse, der KN-Warenkatalog sowie die benutzerdefinierten Warenklassifikationen wurden jeweils als SQLite-Datenbank bereitgestellt. Prinzipiell könnten sämtliche Daten in einer Datenbank verwaltet werden, allerdings wurde aufgrund der leichteren Austauschbarkeit davon abgegangen. Auf die originalen Access-Datenbanken des KN-Warenkatalogs und der benutzerdefinierten Warenklassifikationen konnte über das 64-Bit-basierte JRE nicht zugegriffen werden. Die Daten wurden daher ebenfalls in eine SQLite-Datenbank konvertiert. Die Installation der 64-Bit-basierten Access-Datenbank-Engine würde dieses Problem vermutlich beheben – um nicht unnötige Systemabhängigkeiten und Kompatibilitätsprobleme zu erzeugen, wurden die Daten aber vorerst konvertiert.

Wechselkurse

Die Europäische Zentralbank (EZB) veröffentlicht täglich[10] die Wechselkurse des Euro zu weiteren 33 Währungen im XML-Format. Währungen werden darin mithilfe eines dreistelligen Kürzels laut ISO-4217 codiert. Neben den nationalen Währungen aller EU-Mitgliedsstaaten außerhalb der Eurozone, werden in der Veröffentlichung international bedeutende Währungen berücksichtigt. Die Daten basieren auf dem gewichteten Durchschnitt der Wechselkurse 20 verschiedener Handelspartner außerhalb der Eurozone [17]. Aufgrund der Stellung der EZB innerhalb der EU, kann davon ausgegangen werden, dass es sich um zuverlässige Daten handelt.

Weiters stehen sämtliche Daten seit 1999[11] zur Verfügung. Auf Basis dieser Daten können bei der Erstellung von Intrastat-Meldungen auch die tagesaktuellen Wechselkurse zum Zeitpunkt der Rechnungsausstellung herangezogen werden. Der folgende Code zeigt das XML-Format, in dem die EZB die Daten bereitstellt.

```
<gesmes:Envelope>
  <gesmes:subject>Reference rates</gesmes:subject>
  <gesmes:Sender>
    <gesmes:name>European Central Bank</gesmes:name>
  </gesmes:Sender>
  <Cube>
    <Cube time="2012-10-19">
      <Cube currency="USD" rate="1.3035"/>
      <Cube currency="GBP" rate="0.81275"/>
      <Cube currency="HUF" rate="279.30"/>
      ...
    </Cube>
  </Cube>
</gesmes:Envelope>
```

[10] http://www.ecb.int/stats/eurofxref/eurofxref-daily.xml
[11] http://www.ecb.europa.eu/stats/eurofxref/eurofxref-hist.xml

KN8-Code	*Beschreibung*	*Bes. Maßeinheit*
01	KAPITEL 1: LEBENDE TIERE	
0101	Pferde, Esel, Maultiere und Maulesel, lebend	
0101 21 00	Zuchtpferde, reinrassig	p/st - Anzahl Stück
...		
10	KAPITEL 10: GETREIDE	
1001	Weizen und Mengkorn	
1001 11 00	Hartweizen, Saatgut zur Aussaat	-
...		

Tabelle 5.1: KN-Warenkatalog.

Benutzerdefinierter Warencode	*KN8-Code*	*Warenklassifikation*
Dübel	39259080	Schlagdübel mit Teller
Fräser	82077031	Schaftfräser
...		

Tabelle 5.2: Warenklassifikationen.

KN-Warenkatalog

Auf der Website der Statistik Austria wird der KN-Warenkatalog nur als PDF-Dokument zum Download angeboten. Mithilfe der, ebenfalls bereitgestellten, Software IDEP ist es allerdings möglich, das Warenverzeichnis in diversen Formaten, z.B. als Access-Datenbank, zu exportieren. Die rund 10000 Tarifnummern und übergeordneten Kapiteln sind, wie in Tabelle 5.1 ersichtlich, strukturiert.

Warenklassifikationen

Mithilfe von IDEP kann für jedes meldepflichtige Unternehmen ein angepasstes Warenverzeichnis erstellt werden. Damit werden wiederkehrende Warenklassifikationen verwaltet, um eine effizientere Dateneingabe zu ermöglichen. Gespeichert werden diese Daten in einer Access-Datenbank und bestehen im Wesentlichen aus einer Zuordnung von einem benutzerdefiniertem Warencode zu einem KN8-Code und einer Warenklassifikation. In Tabelle 5.2 ist der Aufbau eines möglichen benutzerdefinierten Warenkatalogs ersichtlich.

Zuordnungen von Warennummern zu Artikeln werden jedoch nicht verwaltet. Somit können keine Rückschlüsse gezogen werden, in welchem Kontext die Klassifikationen bisher eingesetzt wurden. Da ein Warenverzeichnis einem Unternehmen zugeordnet ist, kann einem Artikel des gleichen Unternehmens, mit einer darin definierten Tarifnummer, die gleiche Warenklassi-

fikation als zu überprüfenden Vorschlag ergänzt werden.

5.6 Reasoner

Als Reasoner wird der in Jena integrierte `GenericRuleReasoner` im Forward-Chaining-Mode verwendet. Aus Performancegründen wird auf OWL- oder RDFS-Reasoning verzichtet. Ableitungen erfolgen daher nur, wenn auch entsprechende Regeln formuliert werden.

5.6.1 Ontologie

Im derzeitigen Zustand des Systems kommt der Ontologie im laufenden Betrieb keine größere Bedeutung zu. Das definierte Vokabular stellt dennoch die Grundlage für das Regelsystem dar und wurde auch zum ursprünglichen Aufbau des Triple Stores herangezogen. Während der Entwicklungen wurden laufende Änderungen am Datenschema nicht nachmodelliert, nachdem die Ineffizienz des OWL-Reasonings festgestellt wurde. In Abbildung 5.3 sind daher die ursprünglich geplanten Zusammenhänge zwischen den Klassen dargestellt.

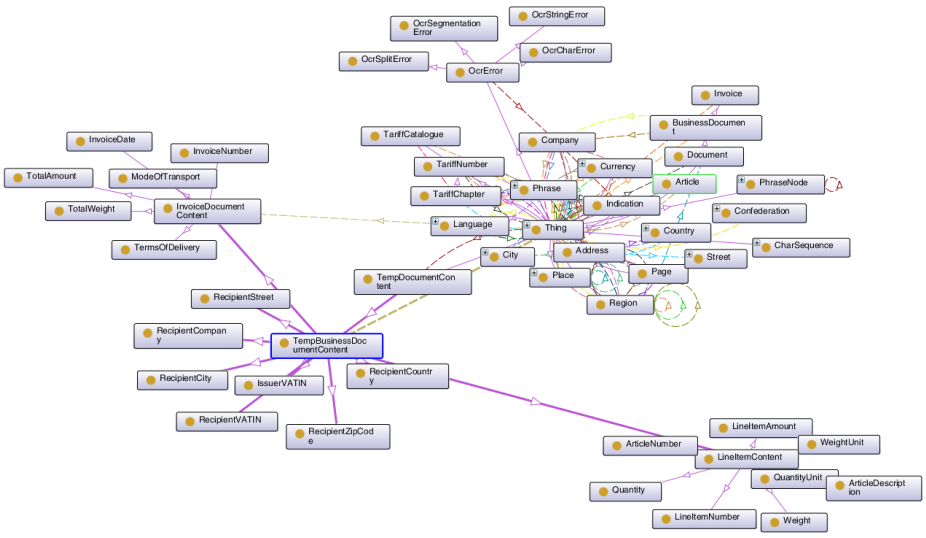


Abbildung 5.3: Zusammenhänge zwischen den Klassen der Ontologie.

5.6.2 Regeln

Jenas integrierte Rule-Engine verwendet eine spezielle Syntax, nach der Regeln zu bilden sind. Wie allgemein üblich, folgen Regeln dem Wenn-Dann-Prinzip. Trifft eine Reihe von Bedingungen zu, werden die entsprechenden

Ableitungen getroffen. Das folgende Beispiel zeigt eine einfache Regel bestehend aus zwei Bedingungen und einer Schlussfolgerung:

```
1 [example:
2   (?x rdf:type    :Company)
3   (?x :hasCountry :Austria)
4   ->
5   (?x rdf:type :AustrianCompany)
6 ]
```

Bedingungen können in Jena nur mit logischem AND verknüpft werden. Regeln, die sich mithilfe eines logischen ORs leichter beschreiben ließen, müssen daher in mehrere Regeln aufgeteilt werden. Der Operator NOT kann ebenso wenig verwendet werden, da nur existierende Fakten geprüft werden können. Es ist erforderlich, auch Negationen explizit abzuleiten.

Neben Variablen stehen eine Reihe von eingebauten Funktionen zur Verfügung. Mithilfe dieser lassen sich beispielsweise Berechnungen oder mathematische Vergleiche durchführen. Weiters kann mit der Funktion `noValue` die Abwesenheit einer Aussage geprüft werden. Bei Verwendung ist jedoch Vorsicht geboten. Da die Ausführungsreihenfolge von Regeln nicht vorhersehbar ist – die Platzierung einer Regel vor einer anderen, ist keine Garantie dafür, dass diese auch zuerst ausgeführt wird – kann die Abwesenheit einer Aussage nur dann als Negation interpretiert werden, wenn keine Regel existiert, die das Statement zur Folge haben könnte.

5.6.3 Erweiterungen

In gewissen Fällen hat es sich als praktisch erwiesen den Funktionsumfang Jenas zu erweitern. Dazu wird von der Klasse `BaseBuiltin` abgeleitet und die erforderlichen Methoden implementiert. Während der Initialisierung des Systems werden die Funktionen bei der Klasse `BuiltinRegistry` registriert. Im Folgenden werden implementierten Erweiterungen vorgestellt.

String-Funktionen

Während andere Regelsprachen, wie z.B. SWRL, eine Reihe von String-Funktionen vorsehen [24], bietet Jena standardmäßig nur eine Funktion zur Verkettung von Strings. Daher wurden folgende Funktionen ergänzt:

- `strLength(?text, ?length)`,
- `strEqual(?text1, ?text2)`,
- `strContains(?text, ?x)`,
- `startsWith(?text, ?x)`,
- `strSplit(?text, ?index, ?partA, ?partB)`.

Mathematische Funktionen

Zur Ermittlung des Absolutwerts einer Variable wurde die Funktion `abs(?x, ?y)` eingeführt. Standardmäßig ist diese Funktion nicht in Jena enthalten, weshalb andernfalls ein Zwischenstatement abgeleitet werden müsste, auf das eine weitere Regel Bezug nimmt. Da keine `elseif`-Konstrukte möglich sind, wären die folgenden zwei Regeln zur Ableitung des Absolutwerts nötig:

```
1 [
2   (?x :hasValue ?y)
3   lessThan(?y, 0)
4   product(?y, -1, ?z)
5   ->
6   (?x :hasAbsValue ?z) ]
7 [
8   (?x :hasValue ?y)
9   ge(?y, 0)
10  ->
11  (?x :hasAbsValue ?y) ]
```

Datentypkonvertierung

Der Standardfunktionsumfang Jenas bietet keine Möglichkeit zur Datentypkonvertierung. Daher wurden die folgenden Funktionen ergänzt:

- `toInt(?text, ?int)`,
- `toDouble(?text, ?locale, ?value)`,
- `toDate(?text, ?format, ?date)`,
- `formatDate(?date, ?format, ?text)`.

Die erste Variable entspricht jeweils dem zu konvertierenden Wert, während die letzte Variable den konvertierten Wert enthält. Die Funktion `toDouble` erwartet als zweiten Parameter den Landescode der `Locale`. Aufgrund der unterschiedlichen Formatierung von Betragsangaben ist diese Angabe nötig, um den Wert korrekt zu interpretieren. Während im deutschsprachigen Raum üblicherweise ein Komma als Dezimaltrennzeichen und ein Punkt als Tausendertrennzeichen verwendet werden, ist es im englischsprachigen Raum umgekehrt. In den folgenden Fällen wird der Variable `value` jeweils der Wert 1000 zugewiesen:

```
1 toDouble('1.000,00', 'de', ?value)
2 toDouble('1,000.00', 'en', ?value)
```

Die Funktionen `toDate` und `formatDate` erwarten als zweiten Parameter die Angabe des Formats und dienen zur Konvertierung zwischen den Datentypen String und Date. Die Variable `date` enthält den numerischen Datenwert und die Variable `formatted` den String `2012-12-14`.

```
1 toDate('14.12.2012', 'dd.MM.yyyy' ?date)
2 formatDate(?date, 'yyyy-MM-dd', ?formatted)
```

Datenbank-Abfragen

Die Einbindung des KN-Warenkatalogs als RDF-Graph in den Reasoning-Prozess hatte, aufgrund der großen Datenmenge, einen negativen Einfluss auf die Performance. Um dennoch die Gültigkeit einer Tarifnummer prüfen zu können, wurde die Funktion `tariffDesc` eingeführt. Intern greift diese Funktion auf den, als SQLite-Datenbank bereitgestellten, Warenkatalog zu und liefert ein positives Resultat, wenn eine abgefragte Tarifnummer existiert. Weiters wird einer zweiten Variable die zur Nummer gehörende Beschreibung zugewiesen. Somit kann die Funktion folgendermaßen verwendet werden:

```
[
  (?x :hasTariffNumber ?num)
  tariffDesc(?num, ?desc)
  ->
  (?x :hasTariffDescription ?desc)
]
```

Auf Basis der Aussage

```
_:x :hasTariffNumber '96091010' .
```

kann somit

```
_:x :hasTariffDescription 'Bleistifte mit festem Schutzmantel' .
```

abgeleitet werden. Intern wird dabei die folgende SQL-Abfrage ausgeführt:

```
SELECT Description FROM Tariff WHERE Number = '96091010';
```

Mit den Funktionen `customTariffDesc` und `exchangeRate` existieren weitere Funktionen um die Beschreibungen des benutzerdefinierten Warenkatalogs und Euro-Wechselkurse via intern durchgeführter Datenbank-Abfragen ins Regelsystem zu integrieren.

Aggregatfunktionen

In relationalen Datenbanksystemen sind Aggregatfunktionen, z.B. zur Berechnung von Spaltensummen, einfach zu verwenden. Die Summe der Rechnungspositionen einer Rechnung lässt sich mit einer einzeiligen SQL-Abfrage ermitteln:

```
SELECT SUM(Amount) FROM LineItems WHERE InvoiceNumber = '00546';
```

Verglichen damit sind Aggregatsfunktionen als Regeln kompliziert zu formulieren. Aufgrund der Graph-basierten Datenstruktur sind dem System keine Spalten ersichtlich, weshalb die Kombination der Daten in mehreren Schritten erfolgen muss. Zuerst wird die Anzahl der Einträge ermittelt. Da dies wiederum die Bildung einer Summe erfordert, wurde die Funktion `countProperties` eingeführt, um die Anzahl der notwendigen Regeln

zu reduzieren und das Verfassen von Regeln zu vereinfachen. Als konkretes Beispiel wird gezeigt, wie die Summe der Rechnungspositionen im System ermittelt wird.

```
[
  (?invoice rdf:type :Invoice)
  countProperties(?invoice, :hasLineItem, ?items)
  ->
  (?invoice :hasNumberOfLineItems ?items)
  (?invoice :checkedLineItems '0'^^xsd:integer)
  (?invoice :hasLineItemSum '0.0'^^xsd:double)
]
```

Neben der Anzahl der Rechnungspositionen, werden zwei weitere Aussagen hinzugefügt. Das Attribut `hasLineItemSum` verweist auf den aktuellen Zwischenstand der Berechnung und `checkedLineItems` enthält die Anzahl der in Berechnung bereits berücksichtigten Positionen. Vorläufig werden diese Werte mit 0 initialisiert. Mit der folgenden Regel wird der Betrag einer Rechnungsposition hinzugerechnet.

```
[
  (?invoice rdf:type :Invoice)
  (?invoice :hasLineItem ?item)
  noValue(?item, :checked)

  (?invoice :checkedLineItems ?numChecked)
  (?invoice :hasLineItemSum ?currentSum)

  (?item :hasTotalAmount ?amount)
  (?amount :hasValue ?value)

  sum(?currentSum, ?value, ?newSum)
  sum(?numChecked, 1, ?newChecked)

  ->
  remove(3)
  remove(4)

  (?invoice :hasCheckedLineItems ?newChecked)
  (?invoice :hasLineItemSum ?newSum)
  (?item :checked 'true'^^xsd:boolean)
]
```

Mit den Funktionen `remove(3)` und `remove(4)` werden die ursprünglichen Werte gelöscht. Dies ist notwendig, da ein bestehendes RDF-Statement nicht verändert werden kann. Ohne die vorherigen Zwischenergebnisse zu löschen, wären mit einer Rechnung jedoch mehrere Zwischensummen verbunden, sodass ein weiteres Konstrukt notwendig wäre, um den aktuellen Stand zu ermitteln.

OCR-Fehlerkorrektur

Die Funktion `tryCorrect` wird verwendet, um einen bestimmten Text, anhand der im System vorhandenen Zeichenersetzungen (`OcrCharErrors`) und einer Menge von gültigen Zeichen zu korrigieren. Die Funktion wird folgendermaßen verwendet:

```
1 tryCorrect(?text, '0123456789', ?corrected)
```

Zusammenfügen von Textteilen

Mit der Funktion `mergeRegions` werden sämtliche Regionen, die mit einer Ressource `node` über die Eigenschaft `predicate` verbunden sind, zusammengefügt. Mit einem `Separator` wird definiert, wie und ob die Textteile voneinander getrennt werden. Die Funktion enthält fünf Ausgabeparameter, die den zusammengefügten Text und die Koordinaten der zusammengefügten Region enthalten.

```
1 mergeRegions(?node, ?predicate, ?separator,
2              ?text, ?top, ?bttm, ?left, ?right)
```

Debugging

Die Funktionen `drawLine`, `drawRect`, `drawRectRGBA` und `logMsg` wurden eingeführt um einzelne Verarbeitungsschritte zu visualisieren bzw. zu protokollieren. Sie bedienen sich des internen Eventsystems um Nachrichten an die zuständigen Komponenten zu senden.

Kapitel 6

Implementierung

In diesem Kapitel werden Details der Implementierung vorgestellt. Die Bilddateien der gescannten Dokumente werden, durch die Kombination unterschiedlicher Wissensquellen, schrittweise in wertvolle Informationen konvertiert. Die Verarbeitung erfolgt dabei in folgenden Schritten:

- XML-Konvertierung,
- Dokumentenklassifizierung,
- OCR-Fehlerkorrektur,
- Schlüsselworterkennung,
- RDF-Konvertierung,
- Regelbasierte Informationsextraktion,
- Manuelle Überprüfung und Nachbearbeitung,
- Datenexport.

6.1 XML-Konvertierung

Zur Durchführung der Texterkennung erwartet Tesseract die Angabe der anzuwendenen Spracheinstellungen. Vor der Konvertierung wird diese manuell durch den Benutzer definiert. Mittlerweile unterstützt Tesseract offiziell über 60 Sprachen und kann bei Bedarf trainiert werden, um an spezielle Anforderungen angepasst zu werden. Auf der Website findet sich dazu eine ausführliche Anleitung[1]. Ein spezielles Training Tesseracts war im Rahmen dieser Arbeit nicht nötig, da Rechnungen mit weit verbreiteten Standardschriftarten gedruckt werden. Bei ausreichender Scanqualität sind mit den Standardeinstellungen Tesseracts gute Ergebnisse zu erzielen. Empfohlen wird eine Auflösung von 300 dpi. Kleingedrucktes, also Text der mit einer Schriftgröße von ca. 8 Punkt oder weniger gedruckt wurde, ist jedoch zunehmend anfällig für OCR-Fehler. Eine niedrigere Auflösung (z.B. 150 dpi) führt, vor allem bei

[1] http://code.google.com/p/tesseract-ocr/wiki/TrainingTesseract3

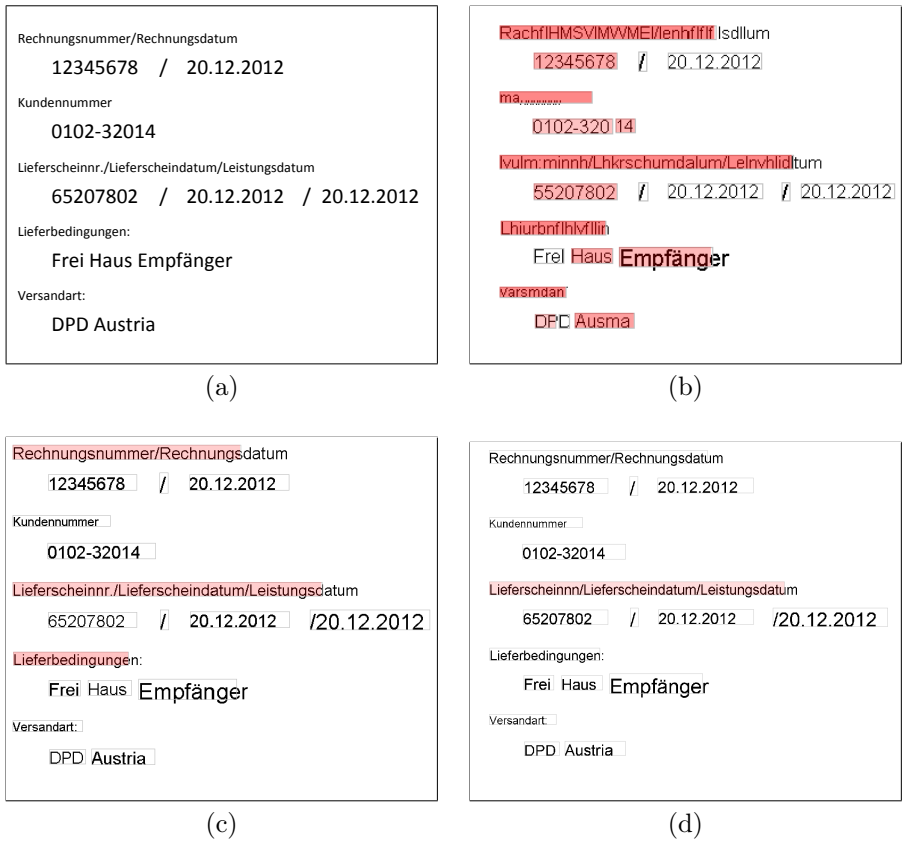


Abbildung 6.1: Ergebnis der Texterkennung bei unterschiedlichen Scanauflösungen. Das Original (a) wurde mit 150 dpi (b), 300 dpi (c) und 600 dpi (d) gescannt.

Kleingedrucktem, zu einer höheren Fehlerquote. Eine höhere Auflösung (z.B. 600 dpi) erbringt keine besseren Resultate, führt jedoch zu einer wesentlichen Verringerung der Performance. In Abbildung 6.1 sind die Qualitätsunterschiede der Texterkennung bei unterschiedlichen Auflösungen ersichtlich. Rot markierte Wörter weisen dabei auf einen niedrigen Zuverlässigkeitswert hin und enthalten mit zunehmender Wahrscheinlichkeit OCR-Fehler.

Abbildung 6.2 visualisiert den Konvertierungsvorgang. Der Ablauf gliedert sich dabei in die folgenden drei Schritte:

- Vorverarbeitung,
- Layoutanalyse und Texterkennung,
- XML-Export.

.

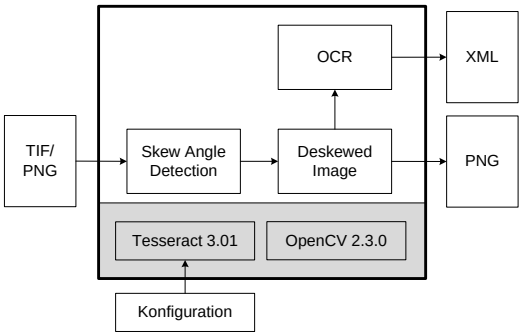


Abbildung 6.2: Aufbau der C++-Anwendung.

6.1.1 Vorverarbeitung

Zunächst wird überprüft, ob ein Dokument schief gescannt wurde und ob eine Korrektur notwendig ist. Dazu wird mithilfe von Tesseract der Verzerrungswinkel (engl. *skew angle*) bestimmt. Bei einem Winkel von mehr als 3° wird die Verzerrung mithilfe von OpenCV korrigiert. Damit ist sichergestellt, dass sich Regionen nach dem Datenexport nicht überschneiden und dass anhand der Koordinaten die relativen Positionen zuverlässig abgeleitet werden können.

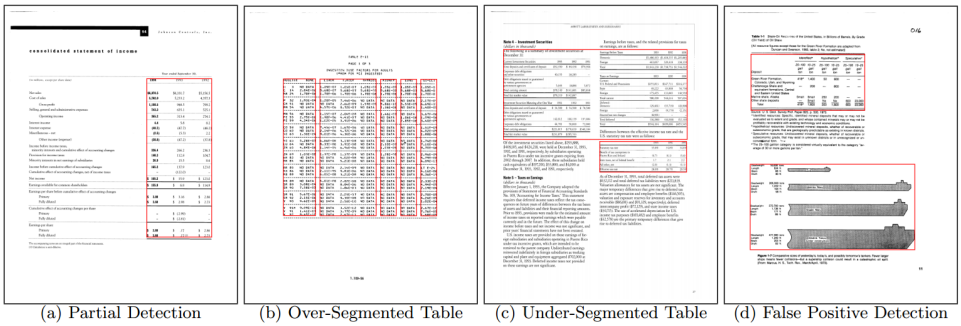
6.1.2 Layoutanalyse und Texterkennung

Tesseract verfügt seit der Version 3.0 über die Möglichkeit zur Layoutanalyse. Der Algorithmus, der zur Analyse des physischen Layouts zur Anwendung kommt, wird in [40] beschrieben. Über die API können Layoutanalyse und Texterkennung über eine Methode aufgerufen werden. Als Ergebnis wird eine Seite in eine hierarchische Struktur, bestehend aus Blöcken, Absätzen, Zeilen, Wörtern und Zeichen, konvertiert. Auf Basis dieser Daten erfolgt der XML-Export.

Logische Layoutanalyse

Die logische Layoutanalyse Tesseracts dient zur Erkennung von logischen Einheiten wie Überschriften, Fließtext oder Tabellen. Aufgrund der tabellenartigen Struktur von Rechnungen, wurde in Erwägung gezogen logische Strukturen ebenso zu exportieren.

In [38] wird Tesseracts Algorithmus zur Tabellenerkennung vorgestellt. Laut den darin präsentierten Ergebnissen wurden, ausgehend von 214 Bildern, 79% aller Tabellen korrekt erkannt. Einige der Fehlerarten sind in Abbildung 6.3 dargestellt. Als häufigste Fehlerursache wurden Tabellen genannt, die eine ganze Seite ausfüllen. Besonders mehrseitige Rechnungen wären potentiell anfällig für dieses Problem. Aufgrund der relativ hohen Fehler-



(a) Partial Detection (b) Over-Segmented Table (c) Under-Segmented Table (d) False Positive Detection

Abbildung 6.3: Fehler bei der Tabellenerkennung (übernommen aus [38]).

quote wurde darauf verzichtet das Ergebnis der logischen Layoutanalyse zu exportieren. Die Segmentierung der Seite erfolgt, unter Anwendung ausstellerspezifischen Wissens, als Teil der regelbasierten Informationsextraktion.

Laut den Release-Notes[2] wurde die aktuelle Version 3.02 allerdings mit einer verbesserten Layoutanalyse ausgestattet. Insofern könnte im Rahmen einer Systemoptimierung eine neuerliche Evaluierung der Tabellenerkennung erfolgen.

6.1.3 XML-Export

Folgende grundlegende Struktur wurde gewählt um die Inhalte einer Seite in ein maschinenlesbares Format zu konvertieren:

```
1 <Page>
2   <Meta>...</Meta>
3   <Content>...</Content>
4 </Page>
```

Eine Seite (`Page`) besteht aus einem Block von Metainformationen (`Meta`) und dem eigentlichen Inhalt (`Content`) eines Dokuments.

Metainformation

In den Metainformationen finden sich Angaben über die, zur `Page` zugehörigen, Bilddatei. Dazu zählen die Höhe und Breite in Pixeln und der Name der Bilddatei. Weiters sind Datum und Uhrzeit der Erstellung sowie die Dauer des Verarbeitungsprozesses enthalten.

```
1 <Meta>
2   <Width>2552</Width>
3   <Height>3484</Height>
4   <Date>06.02.2012</Date>
5   <Time>16:35:23</Time>
6   <ProcessingTime>4.987</ProcessingTime>
```

[2] http://code.google.com/p/tesseract-ocr/wiki/ReleaseNotes

```
  <File>test___001.png</File>
</Meta>
```

Inhalt

Im `Content`-Bereich werden die wesentlichen Bestandteile eines Dokuments zusammengefasst, sodass dieses anhand der exportierten Daten wieder rekonstruiert werden kann. Im Wesentlichen besteht dieser Teil aus einer Auflistung von Linien und Wörtern sowie deren Koordinaten.

Horizontale bzw. vertikale Linien werden als `HorizontalLine` oder `VerticalLine` exportiert. Die Koordinaten des minimal umschließenden Rechtecks (BoundingBox) sind im `Region`-Konstrukt als Attribute enthalten. Eine horizontale Line wird somit folgendermaßen repräsentiert:

```
<HorizontalLine>
  <Region top="1449" left="356" bottom="1472" right="1472" />
</HorizontalLine>
```

Zusammenhängende Zeichenketten werden als `Word` exportiert. In diesem Konstrukt befindet sich die BoundingBox als `Region`, sowie eine Auflistung aller einzelnen Zeichen eines Wortes. Für jedes Zeichen (`Char`) wird ebenfalls die BoundingBox exportiert. Weiters wird der von Tesseract ermittelte Zuverlässigkeitswert (`confidence`) eines Zeichens, sowie, sofern vergügbar, die besten Alternativen und deren Zuverlässigkeitswert, exportiert. Auf Basis dieser Angaben können OCR-Fehler analysiert und dem Benutzer Wörter mit niedrigem Zuverlässigkeitswert als potentiell fehlerbehafteten Text hervorgehoben werden.

Das Wort „Rechnung“ kann folgendermaßen repräsentiert werden (`Alternatives`-Blöcke wurden ab dem zweiten Zeichen entfernt):

```
<Word value="Rechnung">
  <Region top="489" left="280" bottom="549" right="577" />
  <Char value="R" index="0" confidence="87.3926">
    <Region top="490" left="281" bottom="537" right="317" />
    <Alternatives>
      <Char value="H" confidence="80.1283" />
      <Char value="B" confidence="78.9026" />
    </Alternatives>
  </Char>
  <Char value="e" index="1" confidence="89.4054">
    <Region top="503" left="322" bottom="538" right="352" />
  </Char>
  <Char value="c" index="2" confidence="93.542">
    <Region top="503" left="358" bottom="538" right="388" />
  </Char>
  <Char value="h" index="3" confidence="97.887">
    <Region top="490" left="395" bottom="536" right="425" />
  </Char>
  <Char value="n" index="4" confidence="95.4593">
    <Region top="503" left="432" bottom="536" right="464" />
```

```
21   </Char>
22   <Char value="u" index="5" confidence="96.8497">
23     <Region top="504" left="471" bottom="538" right="501" />
24   </Char>
25   <Char value="n" index="6" confidence="95.1816">
26     <Region top="503" left="508" bottom="536" right="540" />
27   </Char>
28   <Char value="g" index="7" confidence="92.4153">
29     <Region top="503" left="546" bottom="549" right="577" />
30   </Char>
31 </Word>
```

6.2 Dokumentenklassifizierung

Unternehmen sind in der Regel bestrebt eine Corporate Identity (CI) aufzubauen. Damit verbunden ist die Entwicklung eines visuellen Erscheinungsbildes, das sich auch in der Gestaltung von Geschäftsdokumenten niederschlägt und dem Unternehmen einen Wiedererkennungswert verleihen soll. Dieser Wiedererkennungswert kann bei der Klassifizierung eines Dokuments genutzt werden. Prinzipiell ist eine Klassifizierung auf verschiedenen Ebenen bzw. nach verschiedenen Verarbeitungsschritten möglich.

Aufgrund der Teilung des Systems in zwei Anwendungen, liegt das vollständige Ergebnis der Texterkennung bereits vor, wenn ein Dokument in die Hauptanwendung importiert wird. Zur Klassifizierung stehen somit das Originalbild, das physische Layout und der enthaltene Text zur Verfügung.

Im deutschsprachigen Raum ist es weitgehend üblich im Seitenfuß eines Geschäftsdokuments sämtliche Geschäftsdaten, wie Unternehmenssitz, Kontaktdaten und Bankverbindungen, anzugeben. Dies beruht auf der DIN 5008 - einer Norm für Brief- und Textgestaltung. Aufgrund der großen Menge an Daten, die sich im Seitenfuß befinden und die eindeutig mit einem Unternehmen verbunden sind, kann die Layoutstruktur und der Inhalt des Seitenfußes genutzt werden um Geschäftsdokumente zuverlässig zu klassifizieren.

Templates

Im derzeitigen Entwicklungsstand sind zur Informationsextraktion manuell verfasste klassenspezifische Regeln nötig. Das Ziel der Klassifizierung ist daher die geeignete Menge von Extraktionsregeln für das aktuelle Dokument zu bestimmen. Dazu wurde ein auf Vorlagen (*Templates*) basierender Algorithmus entwickelt.

Eine Template entspricht einer in XML konvertierten Seite eines Ausstellers. Die Metainformationen werden jedoch um weitere Informationen ergänzt, sodass nach erfolgreicher Klassifizierung, die erfassten OCR-Fehlerkorrekturen (`OcrClass`) und eine Menge von Regeln (`Rules`) zur Informationsextraktion zugeordnet werden kann. Weiters wird die UID-Nummer des

Ausstellers erfasst, da OCR-Fehler und Regeln für mehrere Aussteller verwendet werden können.

```
<Meta>
  ...
  <Issuer>ATU12345678</Issuer>
  <Rules>./rules/<Rules>
  <OcrClass>http://www.example.org#CompanyXY</OcrClass>
  <SplittingChars>/</SplittingChars>
</Meta>
```

Matching

Anhand der Länge der, im Seitenfuß der Vorlage vorhandenen, Wörter wird eine maximal zu erreichende Punktezahl ermittelt. Nach jedem Wort wird im unbekannten Dokument gesucht. Ausgehend von den ursprünglichen Koordinaten wird eine Suchregion erstellt. Da nach dem Scanvorgang die absolute Position abweichen kann, wird die Suchregion an den Rändern um je 1 cm erweitert. Im unbekannten Dokument wird innerhalb dieser Region nach einem Wort gesucht, das am ehesten dem der Vorlage entspricht. Um OCR-Fehler zu berücksichtigen, wird nicht nach exakt dem gleichen Wort gesucht. Der beste Kandidat wird anhand der Levenshtein-Distanz ermittelt. Aus der Levenshtein-Distanz geht hervor wie viele Bearbeitungsschritte notwendig sind, um eine Zeichenkette in eine andere zu transformieren [33]. Die Levenshtein-Distanz zwischen den Wörtern „**R**echnung“ und „**Zei**chnung“ beträgt 2, da der erste Buchstabe ersetzt und ein i eingefügt oder gelöscht werden muss, um ein Wort in das andere umzuformen.

Mithilfe der Levenshtein-Distanz lässt sich also die Ähnlichkeit zweier Zeichenketten bestimmen. Als Minimum wurde eine Überstimmung von 50% mit dem Referenzwort festgelegt. Ist dieses Kriterium erfüllt, wird die Wortlänge, abzüglich der Levenshtein-Distanz, der bisher erreichten Punktezahl hinzugefügt.

Für jedes Wort wird wie beschrieben vorgegangen, sodass aus der erreichten Gesamtpunktezahl der Übereinstimmungsgrad mit der Vorlage hervorgeht. Der minimale Übereinstimmungsgrad wurde mit 60% definiert. Dadurch ist ein ausreichend großer Spielraum für Dokumente gegeben, die nicht in der Sprache des Originaldokuments ausgestellt wurden.

Sämtliche der, zu Testzwecken zur Verfügung stehenden, Rechnungen konnten mit diesem Algorithmus korrekt klassifiziert werden. Es wurde jedoch beobachtet, dass beispielsweise bei italienischen Rechnungen die Angabe der Geschäftsdaten im Seitenfuß weitaus weniger üblich ist, als im deutschsprachigen Raum. Um gegenüber Abweichungen dieser Art flexibel zu sein, müsste der Klassifizierungsalgorithmus geringfügig adaptiert werden. Derzeit kann eine beliebige, in XML konvertierte, Seite eines Ausstellers ohne weitere Nachbearbeitung als Vorlage verwendet werden, da der jeweilige Randbereich beim Matching automatisch extrahiert wird. Dies müsste so ab-

gewandelt werden, dass die Vorlage nur noch die statischen Elemente enthält, die für das Matching relevant sind.

6.3 OCR-Fehlerkorrektur

Die Fehlerkorrektur erfolgt auf Basis der manuell erfassten Fehlerkorrekturen und der definierten `SplittingChars` in der Vorlage des Ausstellers. Die Fehlerkorrektur dient dazu, die nachfolgende Suche nach Schlüsselwörtern zu erleichtern und die Qualität der Ergebnisse zu verbessern. Die Korrektur verläuft in drei Schritten:

1. **Korrektur von Segmentierungsfehlern**: Abhängig von der Schriftart können Segmentierungsfehler entstehen, wenn der Abstand zwischen zwei Zeichen eines Wortes relativ groß ist. Um Fehler dieser Art zu korrigieren, kann ein Abstand definiert werden, bis zum dem benachbarte Textteile zusammengefügt werden. Diese Abstand wird pro `OcrClass` als Zentimeterangabe gespeichert:

```
1 :OcrClass1 rdf:type :OcrClass.
2 :OcrClass1 :hasMaxMergeDistance '0.1'^^xsd:float.
```

2. **Anwendung der `SplittingChars`**: Aus Formatierungsgründen ist möglich, dass mehrere Wörter innerhalb einer Region enthalten sind. Um die Suche nach Schlüsselwörtern zu vereinfachen, werden Regionen, die ein oder mehrere *SplittingChars* enthalten, an den entsprechenden Stellen geteilt. Der Text „Rechnungsnummer/Rechnungsdatum“ würde bei diesem Verarbeitungsschritt in die drei Teile „Rechnungsnummer“, „/“ und „Rechnungsdatum“ geteilt werden. Da die Koordinaten der einzelnen Zeichen als XML mitexportiert wurden, lassen sich die Begrenzungen der neu entstandenen Regionen rekonstruieren.
3. **Anwendung manuell erfasster Fehlerkorrekturen**: Bei der Schlüsselwortsuche wird eine gewisse Fehlertoleranz berücksichtigt, allerdings enthalten kleingedruckte Überschriften häufiger OCR-Fehler. Eine höhere Fehlertoleranz führt wiederum zu häufigeren Fehlklassifikationen, wodurch eine genauere Überprüfung der erkannten Schlüsselwörter innerhalb des Regelsystems notwendig wäre. Fehlerhaft erkannte Wörter können daher über die Benutzeroberfläche korrigiert werden. Bei Bedarf wird die Korrektur dauerhaft zur `OcrClass` gespeichert, wie das folgende Code-Beispiel zeigt:

```
1 :OcrClass1 :hasOcrStringError :OcrStringError1.
2
3 :OcrStringError1 rdf:type :OcrStringError.
4 :OcrStringError1 :hasBadString "BetraQIEUFI)".
5 :OcrStringError1 :hasCorrectString "Betrag(EUR)".
```

   Falls der korrigierte Text die gleiche Länge aufweist als der ursprüngliche Text, werden die ersetzten Zeichen miterfasst:

```
:OcrClass1 :hasOcrCharError :OcrCharError1.

:OcrCharError1 rdf:type :OcrCharError.
:OcrCharError1 :hasBadChar "Q".
:OcrCharError1 :hasCorrectChar "g".
...
```

Bei diesem Verarbeitungsschritt werden Zeichenersetzungen noch nicht berücksichtigt. Im Dokument wird jedoch nach den erfassten `OcrStringErrors` gesucht und mit dem korrekten Text ersetzt.

6.4 Erkennung von Schlüsselwörtern

In tabellarisch strukturierten Dokumenten, wie Rechnungen, ist die Bedeutung von Daten anhand von Überschriften zu erkennen. Eine zuverlässige Erkennung von Schlüsselwörtern bzw. -phrasen ist daher notwendig, um die Daten zu extrahieren. Die Erkennung basiert auf diesen Beobachtungen:

- *Rechnungen sind tabellarisch strukturiert*: Große Freiräume zwischen Wörtern weisen auf getrennte Informationsblöcke hin und können genutzt werden, um die Suche einzugrenzen.
- *Schlüsselphrasen befinden sich innerhalb einer Textzeile*: Es wurden Rechnungen verschiedener Aussteller untersucht, wobei festgestellt wurde, dass sich die relevanten Schlüsselphrasen generell innerhalb einer Textzeile befinden. Da die gesuchten Informationen tabellarisch strukturiert sind und sich nicht innerhalb eines Fließtexts befinden, können Zeilenumbrüche vernachlässigt werden. Eine zeilenübergreifende Suche ist daher nicht notwendig.
- *Mehrere Wörter können eine Phrase bilden*: In deutschsprachigen Dokumenten werden häufig einzelne Wörter, wie z.B. „Rechnungsdatum“ oder „Lieferbedingungen“, als Schlüsselwörter verwendet. Aufgrund der unterschiedlichen Syntax sind in fremdsprachigen Dokumenten zusammengesetzte Phrasen regelmäßig zu finden. Der Suchalgorithmus muss dies berücksichtigen, um Phrasen wie „date of invoice“ oder „terms of delivery“ ebenfalls zu erkennen.
- *Es wird zwischen allgemeingültigen und klassenspezifischen Phrasen unterschieden*: Während sich das Wort „Rechnungsdatum“, unabhängig von der Art des Dokuments und des Kontexts, auf ein Rechnungsdatum bezieht, kann die nähere Bedeutung des Wortes „Datum“ nicht ohne weitere Informationen bestimmt werden. Für einen Rechnungsaussteller, der einen allgemeinen Begriff wie das Wort „Datum“ verwendet um auf das Rechnungsdatum zu verweisen, muss diese Information einmal ins System integriert werden.

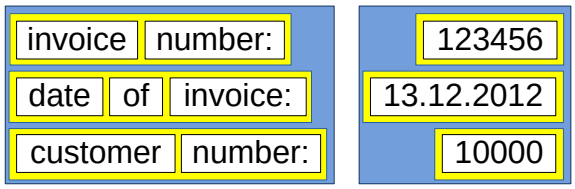


Abbildung 6.4: Segmentierung in Blöcke und Zeilen.

Manuelle Erfassung

Vor der erstmaligen Verarbeitung eines Dokuments werden die Schlüsselphrasen und deren Bedeutung manuell erfasst. Nach allgemeingültigen Phrasen wird in jedem Dokument gesucht. Gespeichert werden diese folgendermaßen:

```
1 :Phrase_Rechnungsdatum rdf:type :Phrase.
2 :Phrase_Rechnungsdatum rdfs:label 'Rechnungsdatum'.
3 :Phrase_Rechnungsdatum :hasIndication :Indication_InvoiceDate.
4
5 :Indication_InvoiceDate rdf:type :Indication.
6 :Indication_InvoiceDate :indicates :InvoiceDate.
```

Klassenspezifische Phrasen enthalten weiters ein `hasClass`-Statement. Nach diesen Phrasen wird nur gesucht, wenn die Klasse mit der `OcrClass` des Templates übereinstimmt.

```
1 :Phrase_Datum rdf:type :Phrase.
2 :Phrase_Datum rdfs:label 'Datum'.
3 :Phrase_Datum :hasIndication :Indication_InvoiceDate2.
4
5 :Indication_InvoiceDate2 rdf:type :Indication.
6 :Indication_InvoiceDate2 :indicates :InvoiceDate.
7 :Indication_InvoiceDate2 :hasClass :ClassXY.
```

Segmentierung

Zunächst werden sämtliche Wörter einer Seite zu Blöcken, bestehend aus Textzeilen, zusammengefasst. Ein Wort wird zu einem Block hinzugefügt, wenn der vertikale und horizontale Abstand zu diesem weniger als 0,5 cm beträgt. Bei üblichen Schriftgrößen von 10 oder 12 Punkt, kann ausgeschlossen werden, dass ein weiter entferntes Wort einer Zeile noch Teil einer eventuellen Schlüsselphrase ist. Dieser Schwellwert ist derzeit fix im Code verankert, könnte jedoch einem Template als weitere Eigenschaft hinzugefügt werden, um die Suche dem Schriftbild einer Dokumentenklasse anzupassen. Das Ergebnis dieses Verarbeitungsschrittes wird in Abbildung 6.4 visualisiert. Blöcke sind blau bzw. Zeilen gelb dargestellt.

Besteht eine Zeile aus mehreren Wörtern, wird geprüft, ob der gesamte Inhalt einer Zeile einer Schlüsselphrase entspricht. Andernfalls wird jedes einzelne Wort getrennt voneinander geprüft. Dieser Algorithmus eignet sich

vorwiegend um Schlüsselphrasen, die im Dokument meist als Überschriften vorkommen, zu erkennen. Um Phrasen innerhalb eines Fließtexts zu erkennen, müsste der Algorithmus noch angepasst werden.

Similarity Score

Unter Berücksichtung von OCR-Fehlern und Formatierungsunterschieden, wird nach einer bekannten Schlüsselphrase gesucht, die am ehesten dem aktuellen Text entspricht. Dazu wird eine Gesamtpunktezahl aus den folgenden drei Parametern errechnet, wobei die Phrase mit der niedrigsten Punktezahl die höchste Übereinstimmung aufweist:

- *Levenshtein-Distanz*: Aus der Levenshtein-Distanz geht die Anzahl der notwendigen Bearbeitungsvorgänge hervor, um einen Text in einen anderen zu transformieren. Darin sind Ersetzungen, Ergänzungen und Löschungen gleichermaßen inkludiert. Eine Distanz von 0 bedeutet eine absolute Überstimmung.
- *Lowercase-Levenshtein-Distanz*: Die Levenshtein-Distanz zwischen den Zeichenketten „Rechnung“ und „RECHNUNG“ beträgt 7, während die Distanz zwischen „Rechnung“ und „Rechnungsdatum“ nur 6 beträgt, obwohl die Übereinstimmung im ersten Fall offensichtlich höher ist. Um diese Formatierungsunterschiede zu berücksichtigen, wird die Levenshtein-Distanz, aufgrund der jeweiligen Lowercase-Entsprechungen, zusätzlich der Punktezahl hinzugerechnet.
- *Textlängenunterschied*: In der Levenshtein-Distanz können Ergänzungs- und Löschvorgänge ebenso enthalten sein, obwohl, unter Berücksichtigung von OCR-Fehlern, Ersetzungsvorgänge als relevanter einzustufen sind. Bei der Texterkennung werden Zeichen zwar verwechselt, allerdings bleibt die Länge des Textes bei guter Scanqualität dennoch erhalten. Der Textlängenunterscheid zweier Phrasen liefert somit einen weiteren Hinweis auf deren Ähnlichkeit.

Bei Phrasen mit einer Länge von bis zu drei Zeichen wird eine Lowercase-Levenshtein-Distanz von 0 erwartet. Bei längeren Phrasen wird die Toleranz dynamisch berechnet, wobei aufgrund von Fehlerkennungen ein Maximalwert von 2 definiert wurde.

Fehlklassifizierungen

Kommt es, aufgrund der Ähnlichkeit zweier Phrasen und der erlaubten Toleranz, dennoch zu fehlerhaften Erkennungen, kann die fehlerhaft klassifizierte Phrase, als Schlüsselwort mit nicht näher definierter Bedeutung, erfasst werden. Während der Entwicklung wurde beispielsweise das Wort „Zeichnung“ häufig als „Rechnung“ interpretiert. Um diese Verwechslung zu vermeiden, wurde „Zeichnung“ als Schlüsselwort mit der Bedeutung `owl:Thing` erfasst.

Intern wird diese Information durch die folgenden RDF-Statements repräsentiert:

```
:Phrase_Zeichnung rdf:type :Phrase.
:Phrase_Zeichnung rdfs:label 'Zeichnung'.
:Phrase_Zeichnung :hasIndication :Indication_Thing.

:Indication_Thing rdf:type :Indication.
:Indication_Thing :indicates owl:Thing.
```

6.5 Konvertierung in RDF-Modell

Als Vorbereitung zur Informationsextraktion wird das Dokument, nach erfolgter Fehlerkorrektur und Schlüsselworterkennung, in einen RDF-Graph konvertiert. Dieser besteht aus einer Seite (`Page`) und ihren Maßen in Pixeln. Nachdem das System auf die Verarbeitung von DIN-A4-Dokumenten ausgelegt ist, können Pixel ins metrische System konvertiert werden. Die Eigenschaft `hasPixelPerCm` wurde hinzugefügt, da bei manchen Regeln der relative Abstand zwischen Wörtern von Bedeutung ist und sich dieser intuitiver durch Zentimeterangaben ausdrücken lässt. Weiters verweist eine Seite auf die ursprüngliche XML-Datei. Die extrahierten Informationen werden später damit verknüpft, um das ursprüngliche Dokument zur Überprüfung darstellen zu können. Über die Eigenschaft `hasRegion` werden alle im Dokument vorkommenden Wörter (*Regions*) einer `Page` zugewiesen. Eine Region besteht aus einem Text und den Koordinaten im Dokument. Schlüsselwörter liefern weiters einen Hinweis auf ein zu extrahierendes logisches Objekt.

Das folgende Beispiel beschreibt eine Seite bestehend aus zwei Regionen:

```
:Page1 rdf:type :Page .
:Page1 :hasWidth '2552'^^xsd:integer .
:Page1 :hasHeight '3484'^^xsd:integer .
:Page1 :hasPixelPerCm '122'^^xsd:integer .
:Page1 :hasXmlFile './xml/0001.xml' .
:Page1 :hasRegion :Region1 .
:Page1 :hasRegion :Region2 .

:Region1 rdf:type :Region .
:Region1 :hasTop '751'^^xsd:integer .
:Region1 :hasBottom '774'^^xsd:integer .
:Region1 :hasLeft '1435'^^xsd:integer .
:Region1 :hasRight '1625'^^xsd:integer .
:Region1 rdfs:label 'Rechnungsnummer' .
:Region1 :indicates :InvoiceNumber .

:Region2 rdf:type :Region .
:Region2 :hasTop '789'^^xsd:integer .
:Region2 :hasBottom '822'^^xsd:integer .
:Region2 :hasLeft '1743'^^xsd:integer .
:Region2 :hasRight '1974'^^xsd:integer .
:Region2 rdfs:label '20.11.2012' .
```

6.6 Regelbasierte Informationsextraktion

Wie im Kapitel „Systemarchitektur“ beschrieben, wird zur Informationsextraktion der `GenericRuleReasoner` des Jena-Frameworks im Forward-Chaining-Mode verwendet. OWL- bzw. RDFS-Reasoning wird nicht vorgenommen, da dadurch eine große Menge von Statements abgeleitet wird, die zur Extraktion nicht unbedingt notwendig ist und die Performance wesentlich verschlechtert. Der Reasoner wendet eine Menge von Regeln auf einen RDF-Graphen an und liefert als Ergebnis wiederum einen RDF-Graphen zurück. Der eingehende RDF-Graph ist das Ergebnis des zuvor durchgeführten Verarbeitungsschrittes. Das `Rules`-Konstrukt im Template verweist auf ein Verzeichnis in dem die anzuwendenden Regeln zu finden sind.

Die Informationsextraktion lässt sich pro Dokument in zwei Phasen einteilen. Zuerst erfolgt eine Verarbeitung der einzelnen Seiten. Aus diesen werden die relevanten Textteile extrahiert und den entsprechenden logischen Objekten zugewiesen. Nach Erkennung der letzten Seite eines Dokuments, erfolgt in der zweiten Phase, unter Einbindung der im Triple Store gespeicherten RDF-Statements, die Ableitung der Intrastat-Daten sowie eine Bewertung der abgeleiteten und extrahierten Informationen.

Um die Komplexität des Systems zu verringern, werden in jeder Phase mehrere verschachtelte Reasoner eingesetzt. Für jede Datei, die sich im Rules-Verzeichnis befindet, wird ein Reasoner instanziert. Auf diese Weise werden die Abhängigkeiten zwischen einzelnen Regeln reduziert und die Notwendigkeit von Überprüfungsregeln verringert.

Phase I

In der ersten Phasen werden die folgenden Schritte durchgeführt:

- Filtern irrelevanter Regionen,
- Bestimmung der Sektorenzugehörigkeit von Regionen,
- Filtern fehlklassifizierter und irrelevanter Schlüsselwörter,
- Ermitteln der relativen Positionen der Regionen zu den Schlüsselwörtern,
- Ermitteln des Dokumententyps,
- Prüfen auf Dokumentenende,
- Erkennen der Rechnungspositionen und den vertikalen Begrenzungen,
- Zuweisen von Textteilen zu logischen Objekten (Key-Value-Beziehungen),
- Zusammenfügen der Textteile pro logischem Objekt,
- Datentypkonvertierungen.

Phase II

Nach Verarbeitung der letzten Seiten, werden in der zweiten Phase diese Schritte durchgeführt:

- Prüfen der Syntax der extrahierten Rechnungsdaten,
- Ableiten der Intrastat-Daten,
- Ergänzung fehlender Daten,
- Verifizierung bzw. Bewertung.

6.6.1 Filtern irrelevanter Regionen

Um die Anzahl der RDF-Statements und damit einhergehend die Verarbeitungszeit zu reduzieren, werden zuerst einige Regionen gefiltert, die aufgrund ihre Lage innerhalb des Dokuments als nicht relevant zu betrachten sind. Dazu zählen Regionen, die sich im Randbereich des Dokuments befinden, denn im Briefkopf und -fuß befinden sich üblicherweise ausschließlich Unternehmensdaten. Um zu diesem Verarbeitungsschritt zu gelangen, musste das Dokument bereits klassifiziert werden. Da neben der UID-Nummer keine weiteren Unternehmensdaten benötigt werden, können diese Regionen gelöscht werden. Wie die folgende Regel zeigt, werden die entsprechenden Regionen zuerst als zu löschend markiert. In einer allgemeinen Löschregel wird auf das `toBeRemoved`-Statement Bezug genommen, um die betroffenen Regionen aus dem System zu entfernen.

```
[
  (?page rdf:type :Page)
  (?page :hasHeight ?h)
  (?page :hasPixelPerCm ?cm)
  product(?h, ?cm, ?max)

  (?page :hasRegion ?reg)
  (?reg  :hasTop ?top)
  lessThan(?top, ?max)
  ->
  (?reg :toBeRemoved 'true'^^xsd:boolean)
]
```

6.6.2 Absolute Positionen

Um die korrekte Positionierung von Regionen zu prüfen bzw. um Regionen anhand ihrer Position zu filtern, werden Regionen einzelnen Sektoren zugewiesen. Wie im Kapitel „Related Work“ beschrieben, verwenden Cesarini et al. neun gleich große Sektoren, wobei die Zuteilung anhand des Mittelpunkts einer Region erfolgt (siehe Abbildung 4.2). Eine Region ist daher immer genau einem Sektor zugeteilt. Ein Schlüsselwort, das sich an der Grenze zwischen den Sektoren N und NE befindet, kann durch geringfügige Positionsabweichungen, von Dokument zu Dokument unterschiedlich zugeordnet

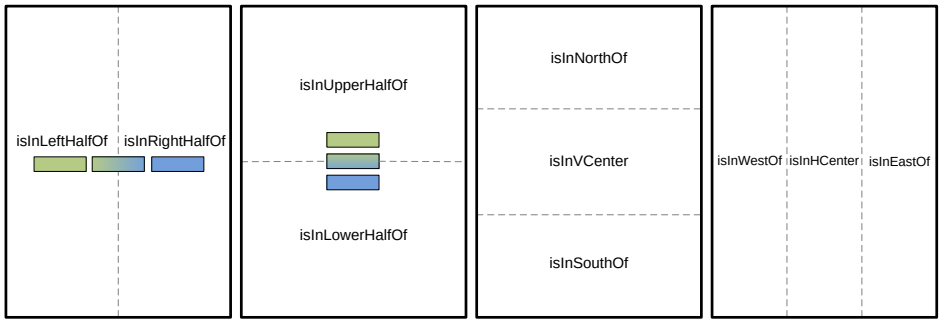


Abbildung 6.5: Einteilung einer Seite in Sektoren.

werden. Weiters bietet die fixe Einteilung einer Seite in neun Sektoren wenig Flexibilität. In manchen Situationen ist es beispielsweise notwendig zu prüfen, ob sich ein Wort in der oberen oder in der linken Hälfte der Seite befindet. Eine Seite wird daher in Hälften und Drittel eingeteilt, wobei die Zugehörigkeit zu einem Sektor dann ist gegeben, wenn sich ein Teil der Region darin befindet. Damit kann eine Region mehreren Sektoren angehören und die Sektorenzugehörigkeit, je nach Bedarf, mit unterschiedlicher Genauigkeit geprüft werden. In Abbildung 6.5 werden die Sektorengrenzen visualisiert.

Im System existiert für jeden Sektor zumindest eine Regel. Mit der folgenden Regel wird geprüft, ob sich eine Region im Sektor *North* befindet.

```
1 [
2   (?page rdf:type :Page)
3   (?page :hasHeight ?h)
4   quotient(?h, 3, ?h1_3)
5
6   (?page :hasRegion ?reg)
7   (?reg  :hasTop ?top)
8
9   lessThan(?top, ?h1_3)
10  ->
11  (?reg :isInNorthOf ?page)
12 ]
```

Je nach Seitenlayout eines Rechnungsausstellers kann es erforderlich sein, die Menge der vordefinierten Sektoren anzupassen oder weitere zu definieren. Generell sollte aus Performancegründen darauf geachtet werden, dass die Zugehörigkeit zu einem Sektor nur dann abgeleitet wird, wenn diese in eine anderen Regel geprüft wird. Für die, während der Entwicklung verwendeten Exemplare, kamen die Sektoren `HCenter` und `VCenter` beispielsweise nicht zum Einsatz.

6.6.3 Relative Positionen

Zur Beschreibung der relativen Position teilen Cesarini et al. die Umgebung einer Region in vier Sektoren ein (siehe Abbildung 4.3). Die Lage zweier Regionen zueinander wird durch exakt eine Beziehung beschrieben. Somit kann sich eine Region nur unterhalb oder nur links von einer weiteren Region befinden. Eine Unterscheidung zwischen links oben und links unten ist nicht möglich. Beide Fälle werden durch die Beziehung `leftOf` beschrieben. Bei der Suche nach Kandidaten ist es jedoch wesentlich, zwischen solchen Fällen unterscheiden zu können. In dieser Arbeit werden folgende Beziehungen zur Beschreibung der relativen Position verwendet:

- `isAboveOf`,
- `isBelowOf`,
- `isLeftOf`,
- `isRightOf`,
- `intersectsVertically`,
- `intersectsHorizontally`.

Um festzustellen, ob sich eine Region über, unter, links bzw. rechts von einer Referenzregion befindet, werden eindimensionale Vergleiche auf Basis der X- bzw. Y-Koordinaten angestellt. Eine Region A befindet sich über einer Region B, wenn $A_{bottom} < B_{top}$. Weiters befindet sich A links von B, wenn $A_{right} < B_{left}$.

Die Beziehungen `intersectsHorizontally` und `intersectsVertically` werden verwendet, um nach Kandidaten zu suchen, die sich in derselben Zeile bzw. Spalte befinden. Eine vertikale Überschneidung findet in folgenden Fällen statt:

- $B_{top} \geq A_{top} \wedge B_{top} < A_{bottom}$,
- $B_{bottom} > A_{top} \wedge B_{bottom} < A_{bottom}$,
- $B_{top} < A_{top} \wedge B_{bottom} > A_{bottom}$.

Analog gelten die gleichen Regeln für eine horizontale Überschneidung:

- $B_{left} < A_{right} \wedge B_{left} \geq A_{left}$,
- $B_{right} > A_{left} \wedge B_{right} < A_{right}$,
- $B_{left} < A_{left} \wedge B_{right} > A_{right}$.

Die Beziehungen sind symmetrisch, weshalb weiters folgende Regeln gelten:

```
1 [ (?x :isAboveOf ?y) -> (?y :isBelowOf ?x) ]
2 [ (?x :isLeftOf  ?y) -> (?y :isRightOf ?x) ]
3 [ (?x :intersectsVertically   ?y) -> (?y :intersectsVertically   ?x) ]
4 [ (?x :intersectsHorizontally ?y) -> (?y :intersectsHorizontally ?x) ]
```

In Abbildung 6.6 werden die Beziehungsvarianten visualisiert.

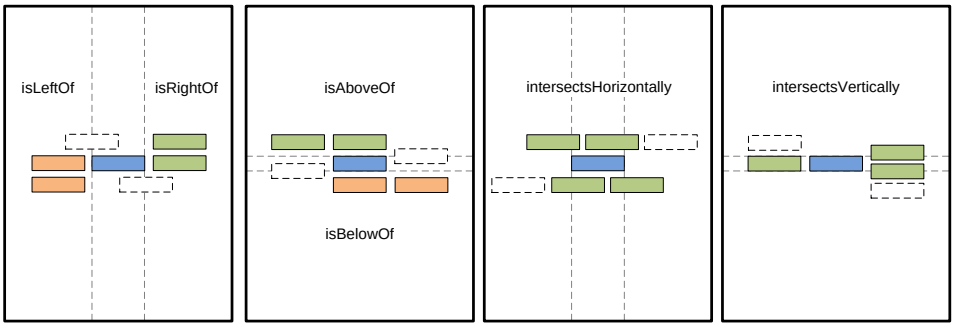


Abbildung 6.6: Relative Positionen.

6.6.4 Ermittlung des Dokumententyps

Anhand der Schlüsselwörter eines Dokuments wird geprüft, ob es sich um eine Rechnung handelt. Es wird bespielsweise erwartet, dass sich im Sektor NW des Dokuments ein Hinweis auf eine Rechnung bzw. im Sektor NE ein Hinweis auf ein Rechnungsdatum oder eine Rechnungsnummer befindet. Um die Auswirkungen von nicht erkannten Schlüsselwörtern gering zu halten, wird die korrekte Position eines Schlüsselworts als ausreichend erachtet. Die folgende Regel zeigt ein einfaches Beispiel für ein derartige Klassifizierung:

```
1 [
2   (?page rdf:type :Page)
3   noValue(?page, :isInvoice)
4   (?reg  :indicates :InvoiceNumber)
5   (?reg  :isInNorthOf ?page)
6   (?reg  :isInEastOf  ?page)
7   ->
8   (?page :isInvoice 'true'^^xsd:boolean)
9 ]
```

Da das System in einer kontrollierten Umgebung eingesetzt wird, wurde keine tiefergehende Art der Klassifizierung implementiert. Es muss jedoch sichergestellt werden, dass die gewählten Bedingungen ausschließlich für die entsprechende Dokumentenart zutreffen. In vielen Fällen sind Geschäftsdokumente so aufgebaut, dass die Annahme zutrifft, dass es sich um eine Rechnung handelt, wenn eines der definierten Schlüsselwörter im Kopf des Dokuments zu finden ist. Je nach Layout und den enthaltenen Schlüsselwörtern, kann es erforderlich sein, genauere Bedingungen zu definieren.

Nachdem der Dokumententyp bestimmt wurde, wird ein leerer RDF-Graph erstellt, der die logische Struktur des Dokuments enthält. Das logische Dokument wird wiederum mit der Seite verknüpft.

6.6.5 Key-Value-Beziehungen

Daten werden aufgrund der relativen Positionen von Regionen zu Schlüsselwörtern extrahiert. Bei Key-Value-Beziehungen besteht innerhalb des Dokumentes eine 1:1-Beziehung zwischen Schlüsselwort und Datenfeld. Ein Datenfeld kann allerdings aus mehreren Regionen bestehen. Segmentierungsfehler können dazu führen, dass einzelne Datenwerte, wie z.B. ein Rechnungsdatum, auf mehrere Regionen verteilt ist.

Die Begrenzung eines Datenfeldes kann durch Absolutwerte, durch die relative Position zu weiteren Schlüsselwörtern oder durch deren Kombination erfolgen. Datenfelder, die numerische Daten enthalten und deren Länge nicht wesentlich variiert, werden durch Absolutwerte begrenzt. Bei Dokumenten mit kleingedruckten Schlüsselwörtern, die für OCR-Fehler besonders anfällig sind, ist es von Vorteil, die erfolgreiche Extraktion nicht von mehr Schlüsselwörtern als nötig abhängig zu machen. Datenfelder mit unterschiedlich langem und möglicherweise mehrzeiligem Inhalt, werden wenn möglich durch weitere Schlüsselwörter abgegrenzt.

Mit der folgenden Regel wird ein Rechnungsdatum anhand von Absolutwerten extrahiert. Regionen, die sämtliche Bedingungen erfüllen, werden dem entsprechenden logischen Objekt dabei als Textteil hinzugefügt. Das Zusammenfügen der Textteile erfolgt erst in einem späteren Schritt.

```
1 [
2   (?num :indicates :InvoiceNumber)
3   (?num :isInNorthOf ?page)
4   (?num :isInEastOf  ?page)
5
6   (?candidate :intersectsVertically ?num)
7   (?candidate :isRightOf ?num)
8
9   (?page :hasPixelPerCm ?cm)
10
11  product(?cm, 0.5, ?minDist)
12  (?num       :hasRight ?numRight)
13  (?candidate :hasLeft  ?regLeft)
14  difference(?regLeft, ?numRight, ?dist)
15  ge(?dist, ?minDist)
16
17  product(?cm, 2.0, ?maxDist)
18  (?candidate :hasRight ?regRight)
19  difference (?regRight, ?numRight, ?dist2)
20  le(?dist2, ?maxDist)
21
22  (?page :hasInvoice ?invoice)
23  (?page :hasInvoiceNumber ?invNum)
24  ->
25  (?invNum :hasTextPart ?reg)
26 ]
```

Die Berechnungen führen allerdings zu einer schlechten Lesbarkeit der Regel.

Im Gegensatz dazu verwendet die nachstehende Regel ausschließlich relative Positionen zur Begrenzung des Datenfeldes.

```
[
  (?terms :indicates :TermsOfDelivery)
  (?terms :isInUpperHalfOf ?page)
  (?terms :isInRightHalfOf ?page)

  (?mode :indicates :ModeOfTransport)
  (?page :hasRegion ?mode)

  (?page :hasRegion ?candidate)

  (?candidate :isBelowOf ?terms)
  (?candidate :isInRightHalfOf ?page)
  (?candidate :isAboveOf ?mode)

  (?page :hasInvoice ?invoice)
  (?invoice :hasTermsOfDelivery ?termsOfDelivery)
  ->
  (?termsOfDelivery :hasTextPart ?candidate)
]
```

6.6.6 Tabellen

Zuerst wird die Anzahl der Zeilen einer Tabelle bestimmt. Bei Rechnungen, die Rechnungspositionsnummern enthalten, kann die Anzahl der Zeilen anhand der Regionen, die sich mit dem Schlüsselwort „Rechnungsposition“ horizontal überschneiden und darunter befinden, bestimmt werden. In der folgenden Regel sind nur die wesentlichen Bedingungen enthalten. Um unzutreffende Regionen auszuschließen, werden Bedingungen ergänzt, welche die erwartete Größe und Lage der Region, sowie den erwarteten Inhalt näher definieren.

```
[
  (?page rdf:type :Page)
  (?page :hasRegion ?itemNumber)
  (?itemHeading :indicates :LineItemNumber)
  (?lineItem :intersectsHorizontally ?itemHeading)
  (?lineItem :isBelowOf ?itemHeading)
  ->
  (?page :hasItem ?lineItem)
]
```

Bei Rechnungen, die über keine Angabe der Positionsnummer verfügen, muss nach einem geeigneten Inhalt gesucht werden, der in jeder Zeile vorhanden ist und anhand dessen die Begrenzung der Zeile abgeleitet werden kann. Je nach Layout können sich Artikelnummern oder -bezeichnungen auch dafür eignen.

Für jede Rechnungsposition wird der logischen Rechnung eine leerer RDF-Graph hinzugefügt, der später mithilfe weiterer Regeln befüllt wird.

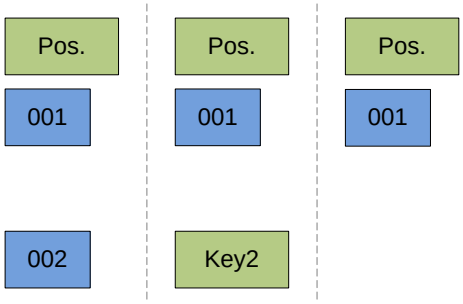


Abbildung 6.7: Übliche Konstellationen.

Vertikale Begrenzungen

Die obere Grenze einer Rechnungsposition geht direkt aus der `hasTop`-Eigenschaft der Rechnungspositionsnummer hervor. Die untere Grenze wird je nach Konstellation abgeleitet. Dabei wird zwischen den folgenden Varianten, die auch in Abbildung 6.7 dargestellt sind, unterschieden:

- Eine Rechnungsposition wird durch eine weitere Position begrenzt. In diesem Fall entspricht die `hasTop`-Eigenschaft der Folgeposition der unteren Grenze.
- Ein Schlüsselwort weist auf das Tabellenende hin. Die `hasTop`-Eigenschaft des Schlüsselwortes kann als untere Grenze einer Position übernommen werden.
- Es ist weder eine Folgeposition noch ein Schlüsselwort vorhanden um die Position vertikal abzugrenzen. Diese Konstellation tritt meist am Ende einer Seite eines mehrseitigen Dokuments auf. In diesem Fall erfolgt die Abgrenzung anhand von Absolutwerten.

Regionen, die sich innerhalb der ermittelten Begrenzungen befinden, können als potentiell relevante Extraktionskandidaten einer Rechnungsposition betrachtet werden. Inhalte einer Position werden über relative Beziehungen zu den, als Schlüsselwörter identifizierten, Spaltenüberschriften extrahiert. Bei unklar strukturierten Tabellen, mit möglicherweise überlappenden Spalten oder mehreren Datenfeldern pro Spalte, kann es erforderlich sein, Regionen zusätzlich untereinander in Beziehung zu setzen. Dadurch ist es möglich die relative Lage einer Region zu einem bereits identifizierten Datenfeld zu bestimmen.

6.6.7 Zusammenfügen der Textteile

Nachdem den logischen Objekten alle relevanten Regionen als Textteile hinzugefügt wurden, werden diese nun zu einem Text zusammengefügt. Wie aus Abbildung 6.8 hervorgeht, ist die Reihenfolge der Textteile zunächst nur implizit in den Koordinaten der Regionen enthalten.

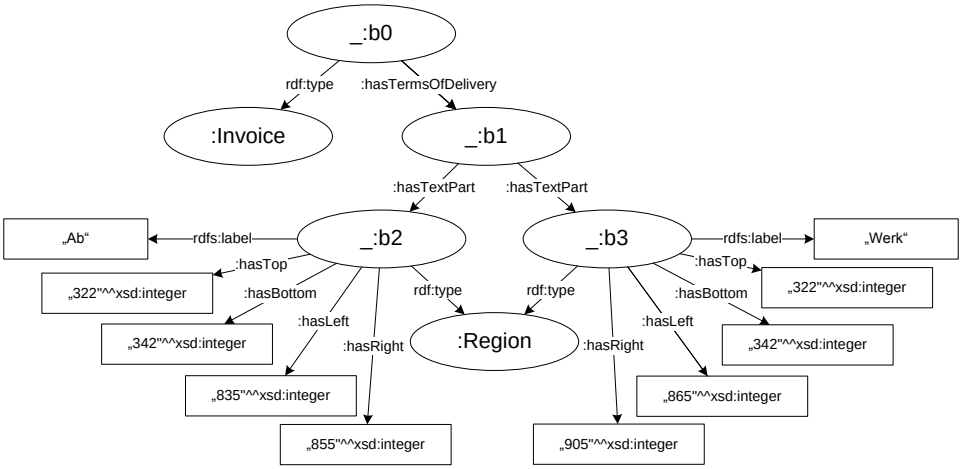


Abbildung 6.8: RDF-Graph vor dem Zusammenfügen der Textteile.

Da sich die Reihenfolge der Textteile mit dem Standardfunktionsumfang Jenas nur mühevoll bestimmen lässt, wird die Funktionserweiterung `mergeRegions` verwendet. Diese berechnet intern die Reihenfolge der Regionen und liefert den zusammengefügten Text und die Koordinaten der entstandenen BoundingBox zurück. Dem logischen Objekt werden diese Informationen hinzugefügt.

```
1 [
2   (?x :hasTextPart ?p)
3   mergeRegions(?x, :hasTextPart, ' ', ?text, ?top, ?bttm, ?left, ?right)
4   makeSkolem(?merged, ?x)
5   ->
6   (?x rdfs:label ?text)
7   (?x :hasRegion ?merged)
8   (?merged :hasTop ?top)
9   (?merged :hasBottom ?bttm)
10  (?merged :hasLeft ?left)
11  (?merged :hasRight ?right)
12 ]
```

6.6.8 Datentypkonvertierungen

Nach Zusammenfügen der Textbestandteile, ist es bei numerischen Daten erforderlich diese zu konvertieren. Dazu werden noch eventuell bestehende OCR-Fehler korrigiert und Formatierungen entfernt.

OCR-Fehler

Die zuvor durchgeführte OCR-Fehlerkorrektur diente hauptsächlich zur Vorbereitung der Schlüsselwortsuche. Einzelne Zeichenersetzungen wurden nicht

durchgeführt und finden erst zum diesem Verarbeitungszeitpunkt statt, da der zu erwartende Datentyp einer Zeichenkette bekannt ist. Durch Definition einer Menge von gültigen Zeichen und der manuell erfassten Zeichenersetzungen innerhalb einer Dokumentenklasse, ist eine gezielte Fehlerkorrektur der extrahierten Zeichenketten möglich.

Beispielweise wurden für das logische Objekt `TotalAmount` die Zeichen „0123456789.,-“ als gültig definiert. Neben sämtlichen Zahlen sind die Zeichen „.“ und „,“ enthalten, da diese als Tausender- bzw. Dezimaltrennzeichen vorkommen. Das Zeichen „-“ wurde wegen möglicherweise negativen Beträgen hinzugefügt. Zudem wurden durch das System die folgenden Zeichenersetzungen, infolge manueller Korrekturen protokolliert:

Fehler	*Korrektur*
O	0
l	1

Auf Basis dieser Daten ist es möglich die Zeichenfolge „4.OO3,ll“ auf „4.003,11“ abzuändern. Die Fehlerkorrektur funktioniert, sofern zu jedem fehlerhaften Zeichen zumindest eine Korrektur manuell erfasst wurde.

Konvertierung

Als nächstes gilt es den Text um die Formatierung zu bereinigen. Dazu werden die Funktionen `toInt`, `toDouble` und `toDate` eingesetzt. Der Endbetrag wird beispielsweise mit der nachstehenden Regel konvertiert.

```
1 [
2   (?invoice :hasTotalAmount ?total)
3   (?invoice rdf:type :Invoice)
4   (?total rdfs:label ?label)
5   toDouble(?label, 'de', ?value)
6   ->
7   (?total :hasValue ?value)
8 ]
```

6.6.9 Extrahiertes RDF-Rechnungsmodell

Nach Durchführung der Fehlerkorrektur sind die Verarbeitungsschritte der ersten Phase abgeschlossen. Als Ergebnis liegt ein RDF-Graph vor, der das logische Modell der extrahierten Rechnung enthält. Das folgende Code-Beispiel zeigt einen vereinfachten RDF-Graph, wie er zu diesem Zeitpunkt vorliegen könnte. Die durch die Regeln erstellten Ressourcen sind nicht durch eine URI referenziert und entsprechen daher BNodes. Jena weist diesen eine eindeutige Identifikationsnummer, z.B. 73e7107:13a74c128d7:-7e48, zu. Um die Lesbarkeit zu erhöhen, wurden diese im folgenden Beispiel umbenannt. Weiters wurde die Verknüpfung mit der physischen Region nur bei einem Objekt angegeben.

```
_:inv1 rdf:type :Invoice .
_:inv1 :hasLineItem _:item0 .
_:inv1 :hasRecipient _:inv1_Recipient .
_:inv1 :hasIssuer _:inv1_Issuer .
_:inv1 :hasInvoiceDate _:inv1_Date .
_:inv1 :hasTotalAmount _:inv1_Total .
...

_:inv1_Recipient rdf:type :Recipient .
_:inv1_Recipient rdfs:label 'DE98765432' .
_:inv1_Recipient :hasRegion _:Region43 .

_:Region43 :isLocatedInFile './sample.xml' .
_:Region43 :hasTop '1049'^^xsd:integer .
_:Region43 :hasBottom '1079'^^xsd:integer .
_:Region43 :hasLeft '1509'^^xsd:integer .
_:Region43 :hasRight '1752'^^xsd:integer .
_:Region43 rdfs:label 'SI12345678' .

_:inv1_Date rdf:type :InvoiceDate .
_:inv1_Date :hasDate '2012-12-01'^^xsd:date .
_:inv1_Date rdfs:label '01.12.2012' .

_:inv1_Total rdf:type :TotalAmount .
_:inv1_Total :hasValue '2620.33'^^xsd:double .
_:inv1_Total rdfs:label '2.620,33' .

_:item0 rdf:type :LineItem .
_:item0 :hasLineItemNumber _:item0_ID .
_:item0 :hasArticleNumber _:item0_ArtNo .
_:item0 :hasArticleDescription _:item0_ArtDesc .
_:item0 :hasWeight _:item0_Weight .
_:item0 :hasWeightUnit _:item0_WUnit .
_:item0 :hasAmount _:item0_Amount .
...

_:item0_ID rdf:type :LineItemNumber .
_:item0_ID rdfs:label '001' .

_:item0_ArtNo rdf:type :ArticleNumber .
_:item0_ArtNo rdfs:label '1005402' .

_:item0_ArtDesc rdf:type :ArticleDescription .
_:item0_ArtDesc rdfs:label 'Schraube 20x8' .

_:item0_Weight rdf:type :Weight .
_:item0_Weight :hasValue '25.3'^^xsd:double .
_:item0_Weight rdfs:label '25,3' .

_:item0_WUnit rdf:type :WeightUnit .
_:item0_WUnit rdfs:label 'KG' .
...
```

6.6.10 Mehrseitige Dokumente

Mehrseitige Dokumente werden so verarbeitet, dass einer Folgeseite das logische RDF-Modell der aktuellen Seite weitergegeben wird. Unter der Voraussetzung, dass Rechnungen sortiert gescannt wurden, ist die Annahme zulässig, dass die Folgeseite zum selben Dokument gehört, wenn in der aktuellen Seite kein Muster enthalten ist, das nur auf der letzten Seite zu finden ist. Im Falle von Rechnungen kann die Tatsache einbezogen werden, dass nach Auflistung der Rechnungspositionen eine Summenbildung erfolgt.

Es kann erforderlich sein die zur Überprüfung eingesetzten Regeln speziell an die Dokumentenklasse anzupassen, da ein zuverlässiges Funktionieren dieser Regeln sichergestellt werden muss. Eine Nicht-Erkennung der letzten Seite führt dazu, dass die Folgeseite als Teil des gleichen Dokuments behandelt wird.

Bisher war es ausreichend, die letzte Seite anhand eines Schlüsselworts, das auf den Endbetrag verweist, zu erkennen. Diese einfache Art der Überprüfung ist allerdings nur möglich, wenn dieses Schlüsselwort nicht für OCR-Fehler anfällig ist und generell zuverlässig erkannt wird.

6.6.11 Intrastat

Nachdem die letzte Seite eines Dokuments erkannt wurde, werden in der zweiten Verarbeitungsphase, auf Basis des extrahierten Rechnungsmodells, die Intrastat-Daten abgeleitet. Dazu wird zuerst die Zielstruktur erstellt. Wie auch bei Rechnungen besteht eine 1:n-Beziehung zwischen allgemeinen Intrastat-Daten und den Einzelpositionen.

In Abbildung 6.9 sind die Zusammenhänge zwischen Rechnungsbestandteilen und Intrastat-Daten anhand einer Versandmeldung veranschaulicht. Grundlage für das Beispiel ist das zuvor gezeigte RDF-Rechnungsmodell.

Vorerst wurde nur der Regelfall endgültiger Transaktionen berücksichtigt. Sonderfälle die sich aufgrund von Dreiecksgeschäften oder Abweichungen zwischen Liefer- und Rechnungsdatum ergeben, wurden nicht berücksichtigt.

Zur Bestimmung der Warenflussrichtung ist vor der Verarbeitung die UID-Nummer des meldepflichtigen Unternehmens durch den Benutzer zu bestimmen. Weiters wurde die KN8-Warennummer nicht zu den Rechnungsbestandteilen gezählt, da diese in Rechnungen nicht unbedingt enthalten sein muss. Bei fehlenden Angaben wird die Klassifizierung anhand der Artikelbezeichnung vorgenommen.

6.6.12 Bewertung der extrahierten Informationen

Bei einer großen Menge von Rechnungen ist eine manuelle Überprüfung der extrahierten Informationen sehr aufwendig, weshalb ein automatische Bewertung vorgenommen wird. Über die Benutzeroberfläche wird die Qualität

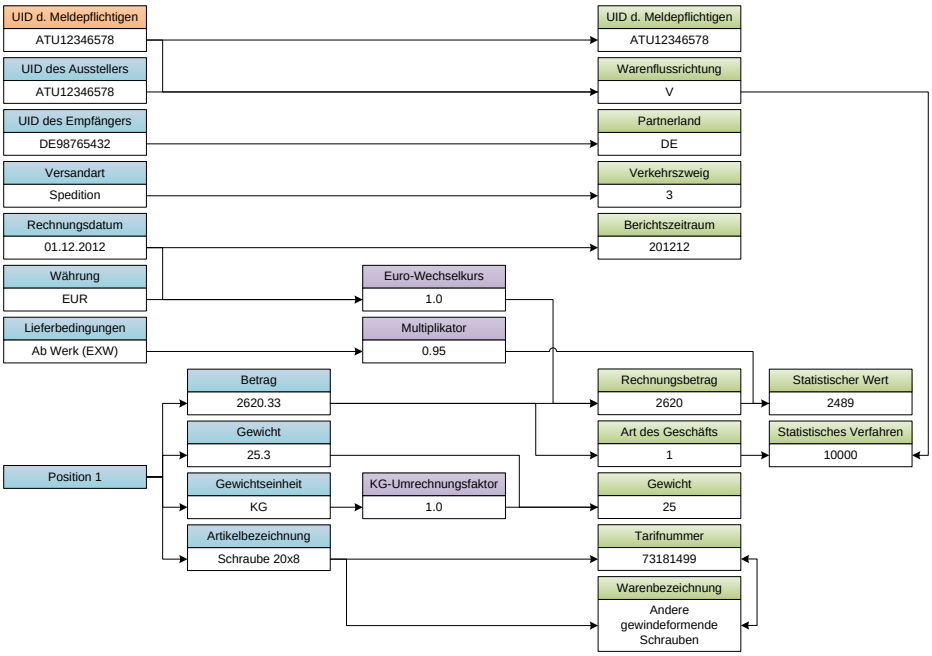


Abbildung 6.9: Zusammenhang zwischen Rechnungsdaten (blau), Benutzereingaben (orange), Zwischenresultaten (violett) und Intrastat-Daten (grün).

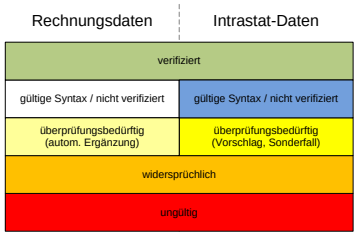


Abbildung 6.10: Farbschema.

der Daten farblich visualisiert. Die Bedeutung der farblichen Codierung wird in Abbildung 6.10 veranschaulicht.

Daten können auf syntaktischer und semantischer Ebene überprüft werden. Auf syntaktischer Ebene wird überprüft, ob die extrahierten Daten der Syntax des erwarteten Datentyps entsprechen. Syntaxfehler und fehlende Daten, die nicht automatisch ergänzt werden können, werden als ungültig bewertet. Ein syntaktisch korrekter Wert ist die Voraussetzung für eine weitere semantische Überprüfung. Informationen werden entweder als *verifiziert*, *überprüfungsbedürftig* oder *widersprüchlich* klassifiziert.

Verifizierte Informationen

Informationen gelten als verifiziert, wenn diese mithilfe der im Triple Store gespeicherten Fakten bestätigt werden können. Diese Fakten wurden zuvor manuell durch einen Benutzer überprüft und können als korrekt betrachtet werden. Um gegenüber minimalen OCR-Fehlern robust zu sein, wird bei textuellen Daten eine gewisse Fehlertoleranz eingeräumt. Die erlaubte Abweichung bei numerischen Daten wird, abhängig vom jeweiligen logischen Objekt, definiert. Während das Gewicht eines Artikel üblicherweise gleichbleibend ist, können sich Artikelpreise, aufgrund von allgemeinen Preisänderungen oder kundenspezifischen Preisen, von Rechnung zu Rechnung durchaus unterscheiden.

Überprüfungsbedürftige Informationen

Informationen werden als überprüfungsbedürftig eingestuft, wenn diese anhand von potentiell unsicheren Fakten abgeleitet wurden, wenn diese als Vorschlag ohne direkten Hinweis ergänzt wurden oder wenn diese nicht dem Regelfall entsprechen.

Widersprüchliche Informationen

Extrahierte Informationen werden als widersprünglich eingestuft, wenn diese nicht mit den bereits verifizierten Fakten übereinstimmen bzw. wenn eine zu hohe Abweichung zu diesen besteht. Die Informationen müssen nicht unbedingt falsch sein, sollten bei der manuellen Überprüfung mit höherer Priorität behandelt werden. Widersprüchliche Informationen entstehen infolge von OCR-Fehlern oder aufgrund von nicht erfassten Änderungen ausstellerspezifischer Daten.

6.7 GUI

Das GUI kann im Wesentlichen als Editor zur Bearbeitung von Literalen eines RDF-Graphs betrachtet werden. Während die Daten, wie in relationalen Datenbanken, in Form einer 1:n-Beziehung zwischen allgemeinen Rechnungsdaten und Rechnungspositionen, präsentiert werden, bleibt im Hintergrund die graphbasierte Datenstruktur erhalten. Mit jedem Eingabefeld ist eine Ressource assoziiert. Bei numerischen Daten wird der Wert der `hasValue`-Eigenschaft bearbeitet bzw. bei textuellen Daten der Wert der `rdfs:label`-Eigenschaft. In Abbildung 6.11 ist der Zusammenhang zwischen GUI-Komponenten und dem extrahierten RDF-Graph grafisch dargestellt.

Zur leichteren Überprüfung der Daten wird bei Auswahl einer Rechnung im linken Bereich das gescannte Originalbild angezeigt. Die zuvor bestimmten vertikalen Begrenzungen der Rechnungspositionen können genutzt werden, um den entsprechenden Bereich bei der Auswahl hervorzuheben. Jedes

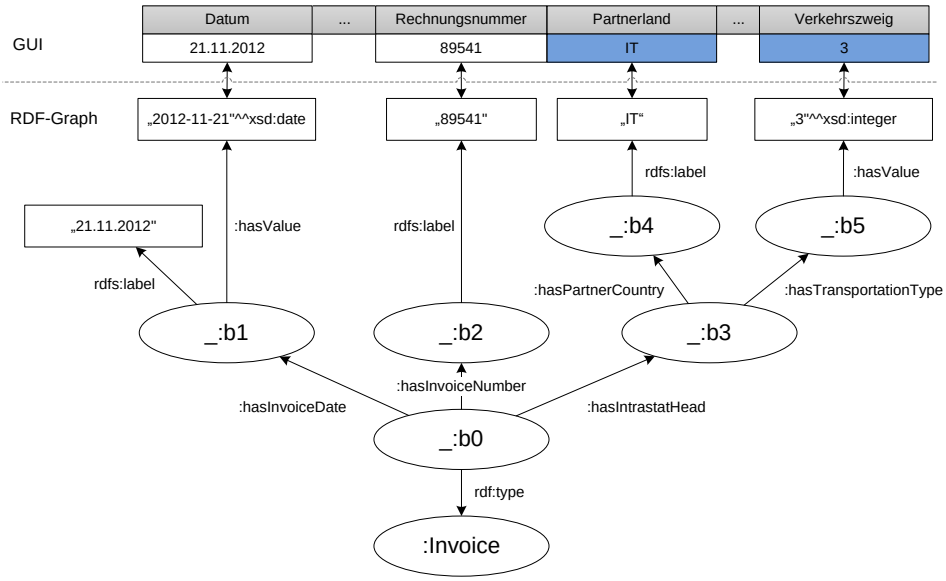


Abbildung 6.11: Zusammenhang zwischen GUI-Komponenten und extrahiertem RDF-Graph.

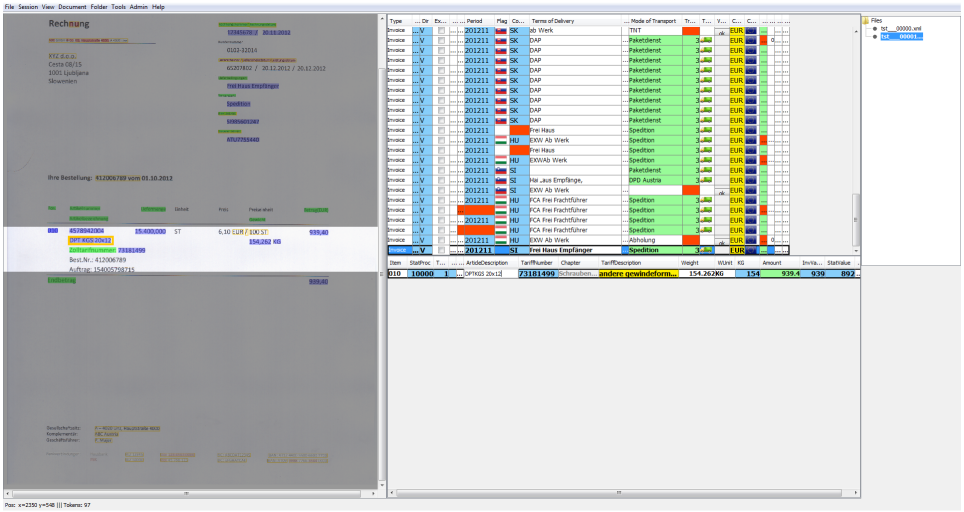


Abbildung 6.12: GUI.

logische Objekt, das aus dem Dokument extrahiert wurde, enthält einen Verweis auf die physische Region innerhalb des Dokuments, aus dem die Daten stammen. Dieser Zusammenhang wird bei der Bearbeitung ebenfalls visualisiert, indem der Bereich hervorgehoben wird. In Abbildung 6.12 ist ein Screenshot der entwickelten Anwendung zu sehen.

6.8 Datenexport

Nach benutzerseitiger Überprüfung und Bearbeitung, können die erstellten Intrastat-Daten in Form einer Excel-Datei aus dem System exportiert werden. Es ist in weiterer Folge vorgesehen diese Daten über die Software IDEP an die Statistik Austria zu versenden. Prinzipiell wäre es, wie in [4] beschrieben, möglich Intrastat-Meldungen direkt aus dem System, via E-Mail oder Upload auf einen FTP-Server, zu versenden. Allerdings ist dazu eine weitere Konvertierung in das entsprechende EDIFACT-Format nötig. Eine detaillierte Beschreibung dieses Formats wird in [5] gegeben. Ein häufiger Fehler versandter Meldungen ist laut [4] die Nicht-Erfüllung formaler Kriterien des definierten Formats. Auf eine Implementierung der Versandfunktion wurde verzichtet, da IDEP über diese Funktion bereits verfügt, die Software frei verfügbar ist und sämtliche Intrastat-Daten weiterhin zentral mithilfe einer Anwendung verwaltet werden können. Weiters erfolgt während des Datenimports eine Fehlerprüfung durch IDEP. Auf diese Weise können eventuelle Exportfehler rechtzeitig vor dem Versand erkannt und korrigiert werden.

Kapitel 7

Evaluierung

In diesem Kapitel werden die Ergebnisse der Evaluierung präsentiert. Dabei wird gezeigt, wie erfolgreich Informationen, mithilfe des vorgestellten Ansatzes, extrahiert werden. Weiters werden die Fehlerquellen analysiert und abschließend Verbesserungsmöglichkeiten des Systems diskutiert.

7.1 Informationsextraktion

Das System wurde anhand von 89 Rechnungen eines Unternehmens evaluiert. Die Rechnungen sind auf insgesamt 162 Seiten verteilt und enthalten 104 Rechnungspositionen. In fünf Fällen werden Dienstleistungen verrechnet, die keiner Meldepflicht unterliegen. Als Ergebnis sollen insgesamt 99 Intrastat-Datensätze erstellt werden. Gescannt wurden sämtliche Seiten mit einer Auflösung von 300 dpi, um für eine ausreichende Bildqualität zu sorgen. Technische Details zur Testumgebung sind der folgenden Tabelle zu entnehmen:

Hardware:	Intel Core i7-2630QM @ 2.0 GHz 8 GB RAM
Betriebssystem:	Windows 7 64-Bit
Java:	Version 6 Update 26 Java SE Runtime Environment (build 1.6.0_26-b03) Java HotSpot 64-Bit Server VM (build 20.1-b02, mixed mode)

7.1.1 Erfasste Daten

Wie in den vorangehenden Kapiteln erwähnt, werden Dokumente klassenspezifisch verarbeitet. Schlüsselwörter, sowie eine Reihe typischer OCR-Fehler, wurden zuvor manuell erfasst und der Dokumentenklasse zugeordnet. Es wurden 17 Schlüsselwörter definiert und, zur besseren Erkennung derselben,

24 OcrStringErrors erfasst. Davon wurden einige exemplarisch ausgewählt um einen Eindruck der OCR-Fehler zu vermitteln:

Schlüsselwort	*OcrStringError*
Rechnungsnummer	Rochnungsnumnter
Rechnungsdatum	Rechnungsdaturrl
Lieferbedingungen	Liembedmgungen
Versandart	Veßandam
Betrag(EUR)	BetraQIEUFI)
	BetragiEUFli
	ßetragfEURl
Ust.-Id.	UsL-ld.
	USIÄd

Zudem wurden zur datentypbasierten Fehlerkorrektur einige Zeichenersetzungen erfasst. Dazu zählen und anderem die Folgenden:

Falsch	*Korrekt*
O	0
i	1
l	I
3	S

7.1.2 Ergebnisse

Die Verarbeitung der 162 Seiten betrug im Hauptsystem rund 20 Minuten. Im Durchschnitt wurde eine Seite mit 7,4 Sekunden verarbeitet.

Dokumentenklassifizierung

Zur Klassifizierung wurden 16 Templates unterschiedlicher Firmen bereitgestellt, wobei sämtliche Seiten korrekt klassifiziert wurden. Durchschnittlich wurde eine 90%-ige Überstimmung mit dem Template ermittelt. Dabei wurden ein Minimalwert von 82% und Maximalwert von 95% erreicht.

Rechnungsdaten

Aus Tabelle 7.1 geht hervor, wie zuverlässig und korrekt einzelne Rechnungsbestandteile extrahiert wurden. In manchen Rechnungen sind bestimmte Angaben nicht enthalten, weshalb die Summe der Vorkommnisse einzelner Bestandteile teilweise variiert. Weiters wurden extrahierte Artikelbeschreibungen auch dann als fehlerlos gewertet, wenn geringfügige OCR-Fehler enthalten sind, die sich aber nicht wesentlich auf den weiteren Vorgang auswirken. In allen weiteren Fällen führte ein fehlerhaft extrahiertes Zeichen dazu, dass das Ergebnis nicht als korrekt bewertet wurde.

Objekt	*Anzahl*	*Erkannt*	*Prozent*	*Korrekt*	*Prozent*
Rechnungsnummer	89	83	93%	83	100%
Rechnungsdatum	89	85	96%	77	91%
UID des Ausstellers	89	89	100%	89	100%
UID des Empfängers	89	85	96%	75	88%
Lieferbedingungen	89	84	94%	77	89%
Versandart	87	85	98%	85	100%
Endbetrag	88	88	100%	82	93%
Rechnungspositionen	104	104	100%	104	100%
Positionsnummer	104	104	100%	101	97%
Artikelnummer	104	104	100%	103	99%
Artikelbeschreibung	104	99	95%	83	83%
Tarifnummer	99	99	100%	99	100%
Gewicht	99	90	91%	86	96%
Gewichtseinheit	99	87	88%	81	93%
Positionsbetrag	104	84	81%	82	98%
Summe	*1437*	*1370*	*95%*	*1307*	*95%*

Tabelle 7.1: Ergebnisse der Informationsextraktion

Generell besteht eine Erkennungsrate von 95%, wobei bei 1437 Objekten 67 Erkennungsfehler auftraten. 35 dieser Fehler sind auf nicht erkannte Schlüsselwörter zurückzuführen. Die restlichen 32 Fehler sind entstanden, da diverse Layoutvariationen nicht durch die vorhandenen Regeln abgedeckt wurden.

Von den insgesamt 1370 erkannten Objekten wurden die Daten in ebenfalls 95% der Fälle korrekt extrahiert. 63 Extraktionsfehler setzen sich aus 41 OCR-Fehlern und 22 Regel-Fehlern zusammen. Letztere sind ebenfalls auf unberücksichtigte Layoutvariationen zurückzuführen. Extraktionsfehler lassen sich hinsichtlich ihrer weiteren Auswirkung als *bedeutend* oder *unbedeutend* klassifizieren. Unbedeutende Fehler wirken sich auf den weiteren Verlauf nicht aus und enthalten nur geringfügige OCR-Fehler. Bedeutende Fehler erfordern eine manuelle Überprüfung, Korrektur oder Ergänzung. Ein bedeutender Fehler wird weiters als *scherwiegend* betrachtet, wenn dieser zu ausbleibenden oder fehlerhaften Ableitungen von Intrastat-Daten führt. Im Rahmen der Evaluierung wurden 5 unbedeutende und 58 bedeutende Fehler festgestellt. In 30 Fällen handelte es sich um schwerwiegende Fehler.

Intrastat-Daten

Das Ergebnis der abgleiteten Intrastat-Daten ist in Tabelle 7.2 zusammengefasst. Auf Basis der 162 Seiten wären insgesamt 1049 Intrastat-Daten ableitbar. Nach der Verarbeitung liegen 89% dieser Daten in korrekter Form vor.

Objekt	*Anzahl*	*Erkannt*	*Prozent*	*Korrekt*	*Prozent*
Handelsrichtung	89	89	100%	89	100%
Zeitraum	89	77	87%	77	100%
Partnerland	89	75	84%	75	100%
Verkehrszweig	89	79	89%	79	100%
Stat. Verfahren	99	99	100%	99	100%
Art des Geschäfts	99	99	100%	82	83%
Tarifnummer	99	99	100%	99	100%
Warenklassifikation	99	97	98%	97	100%
Gewicht	99	87	88%	86	99%
Betrag	99	99	100%	78	79%
Stat. Wert	99	91	92%	70	77%
Summe	*1049*	*991*	*94%*	*931*	*94%*

Tabelle 7.2: Ergebnis der Intrastat-Konvertierung.

Sofern zuvor die entsprechenden Rechnungsdaten extrahiert werden konnten, war eine korrekte Klassifizierung in 94% der Fälle möglich. 112 fehlende oder fehlerhafte Intrastat-Daten sind die direkte Folge fehlender oder fehlerhafter Rechnungsdaten. In den übrigen sechs Fällen konnten korrekt extrahierte, aber dem System unbekannte, Daten nicht klassifiziert werden.

Zusammenfassung

Die Konvertierung von Rechnungs- in Intrastat-Daten erfolgt zuverlässig, sofern erstere in korrekter Form vorliegen. Der manuelle Nachbearbeitungsbedarf, der für 11% der Intrastat-Daten besteht, entsteht zum wesentlichen Teil durch unkorrigierte OCR-Fehler und ungenau definierte Regeln. Während unberücksichtige Layoutvariationen ein klassenspezifisches Problem darstellen, ist die zuverlässige Korrektur von OCR-Fehlern von zentraler Bedeutung für sämtliche Dokumentenklassen. Mithilfe manuell erfasster OcrStringErrors konnte die Fehlerquote bereits reduziert werden, allerdings werden Fehler erst nach dem erstmaligen Auftreten erfasst. Weiters stellt die Erfassung der vielen fehlerbehaften Varianten der einzelnen Wörter einen beträchtlichen Aufwand dar. Diese Form der Fehlerkorrektur kann daher als reaktiv betrachtet werden und müsste in ein proaktive Form umgewandelt werden. Die Weiterentwicklung des Algorithmus zur Schlüsselworterkennung ist daher von essentieller Bedeutung um die Fehlerquote, über Dokumentenklassen hinweg, zu reduzieren.

7.2 Verbesserungspotential

7.2.1 Schlüsselworterkennung

Wie zuvor erwähnt, bietet der Algorithmus zur Schlüsselworterkennung noch Verbesserungspotential. Um eine bessere Erkennung zu erzielen, könnten die bisher erfassten Zeichenersetzungen berücksichtigt werden. Somit könnte, nach einer ersten negativen Überprüfung, noch versucht werden einzelne Zeichen, entsprechend der erfassten Fehler, zu ersetzen. Wird statt des Schlüsselwortes „Betrag(EUR)“ die Zeichenkette „ßetraQiEURi“ erkannt, ließe sich das Schlüsselwort anhand der bekannten Fehler

$$\begin{aligned} \text{ß} &\rightarrow \text{B} \\ \text{Q} &\rightarrow \text{g} \\ \text{i} &\rightarrow \text{(} \end{aligned}$$

rekonstruieren. Da Tesseract jedes Zeichen mit einem Zuverlässigkeitswert versieht, wurde analysiert, inwieweit sich dieser als Indikator zur Erkennung fehlerhafter Zeichen verwenden ließe. Als Grundlage wurden die Zuverlässigkeitswerte 74 fehlerhaft erkannter Zeichen herangezogen. Der Wertebereich erstreckt sich von 52 bis 99, wobei ein Durchschnittswert von 87 ermittelt wurde.

Weiters stellte sich heraus, dass 47% der falsch erkannten Zeichen mit einem Zuverlässigkeitswert von über 90 eingestuft wurden. 31% der Zeichen hatten einen höheren Wert als 95 und 16% einen Wert über 98. Aus den Ergebnissen geht deutlich hervor, dass anhand der Zuverlässigkeitswerte kein direkter Rückschluss auf einen OCR-Fehler gezogen werden kann.

7.2.2 Parallelisierung

Da die einzelnen Seiten derzeit sequentiell abgearbeitet werden, besteht hierbei noch erhebliches Potential die Verarbeitungsdauer einer Seite zu reduzieren. Die regelbasierte Informationsextraktion kann als Engpass betrachtet werden, da der Großteil der Verarbeitungsdauer dieser zuzuschreiben ist. Durch Parallelisierung könnten sämtliche Vorgänge, von der Dokumentenklassifizierung bis zur RDF-Konvertierung, in einem weiteren Thread vorbereitet werden, um die Zeitspanne zwischen Ausführen des Reasoners gering zu halten.

7.2.3 Optimierung der Regeln

Den Großteil der Verarbeitungszeit beansprucht der Reasoner. Aus diesem Grund wurde analysiert, wodurch die Performance des Reasoners gesteigert werden kann.

Anhand eines Dokuments, bestehend aus 244 Regionen und 17 Schlüsselwörtern, wurde überprüft, ob die Reihenfolge der Bedingungen die Verarbeitungszeit beeinflusst. Um eine gezielte Beobachtung zu ermöglichen, wurde die Menge der Regeln reduziert, sodass nur die relativen Beziehungen zwischen den Regionen einer Seite abgeleitet wurden. Dabei wurden zwei Varianten miteinander verglichen, die sich nur durch die Anordnung der ersten beiden Bedingungen unterscheiden.

Variante A	Variante B

`[`
`  (?page :hasRegion ?r1)`
`  (?r1 :indicates ?ind)`
 	
`  (?page :hasRegion ?r2)`
 	
`  (?r1 :hasBottom ?bttm1)`
`  (?r2 :hasTop ?top2)`
`  lessThan(?bttm1, ?top2)`
 	
`->`
 	
`  (?r1 :isAboveOf ?r2)`
`  (?r2 :isBelowOf ?r1)`
`]`
	`[`
	`  (?r1 :indicates ?ind)`
	`  (?page :hasRegion ?r1)`
	
	`  (?page :hasRegion ?r2)`
	
	`  (?r1 :hasBottom ?bttm1)`
	`  (?r2 :hasTop ?top2)`
	`  lessThan(?bttm1, ?top2)`
	
	`->`
	
	`  (?r1 :isAboveOf ?r2)`
	`  (?r2 :isBelowOf ?r1)`
	`]`

Um vergleichbare Werte zu erzielen, wurde dasselbe Dokument je zehn Mal mit Variante A bzw. B verarbeitet. Die durchschnittliche Verarbeitungsdauer beträgt 5205 ms bei Variante A und 4306 ms bei Variante B. Der Unterschied von rund 900 ms legt nahe, dass die Anordnung der Bedingungen die Performance stark beeinflussen kann. Die wesentlichen Bedingungen einer Regel sollten daher vor allen weiteren angeführt werden.

Bei dieser Untersuchung wurde ein einfaches Beispiel herausgegriffen – bei der Verarbeitung eines Dokuments kommt jedoch eine Menge weiterer Regeln zum Einsatz, sodass durch nicht optimierte Reihungen, einige Sekunden pro Seite verloren gehen können. Das Optimierungspotential der Extraktionsregeln wurde noch nicht völlig ausgeschöpft, sodass die zuvor ermittelte Verarbeitungsdauer einer Seite von 7,4 Sekunden noch deutlich reduziert werden kann.

7.2.4 Alternatives Regelsystem

Ein Vorteil des Regelsystems Jenas ist die einfache Erweiterbarkeit um Funktionen. Dennoch wäre ein Repräsentationssystem mit höherer Ausdrucksstärke wünschenswert. Mit der SPARQL Inferencing Notation (SPIN) existiert

eine vielversprechende Alternative. SPIN bietet eine Syntax um SPARQL-Abfragen als RDF-Graph zu speichern [30]. SPARQL ist eine Abfragesprache für RDF-Graphen und unterstützt unter anderem Disjunktionen, Negationen und Aggregationen. Mit CONSTRUCT-Abfragen können Fakten auf Basis von Bedingungen erstellt werden [23]. Mithilfe von SPIN lassen sich solche CONSTRUCT-Abfragen auch als Regel repräsentieren, wodurch die volle Ausdrücksstärke SPARQLs zur Verfügung steht. Zur Formulierung von Regeln wird das SPIN Modeling Vocabulary genutzt [29]. Das folgende Beispiel wurde aus [29] übernommen und zeigt, wie sich die SPARQL-Abfrage

```
1  CONSTRUCT {
2    ?this ex:grandParent ?grandParent .
3  }
4  WHERE {
5    ?parent ex:child ?this .
6    ?grandParent ex:child ?parent .
7  }
```

als SPIN-Regel, unter Verwendung der Turtle-Syntax, ausdrücken lässt.

```
1  ex:Person
2      a rdfs:Class ;
3      rdfs:label "Person"^^xsd:string ;
4      rdfs:subClassOf owl:Thing ;
5      spin:rule
6              [ a sp:Construct ;
7                sp:templates ([ sp:object sp:_grandParent ;
8                                sp:predicate ex:grandParent ;
9                                sp:subject spin:_this
10                             ]) ;
11               sp:where ([ sp:object spin:_this ;
12                           sp:predicate ex:child ;
13                           sp:subject sp:_parent
14                         ] [ sp:object sp:_parent ;
15                           sp:predicate ex:child ;
16                           sp:subject sp:_grandParent
17                         ])
18             ] .
```

Während der Recherche konnte mit der SPIN API[1] nur eine Implementierung dieser Sprache gefunden werden. Diese basiert auf Jena und wäre somit potentiell kompatibel zum entwickelten System. Wie aus der Dokumentation hervorgeht, ist die SPIN API ein Produkt der Firma TopQuadrant[2]. Um den vollen Funktionsumfang nutzen zu können, ist eine kommerzielle Lizenz erforderlich. Weiters sind die Basisfunktionen nur unter der eingeschränkten AGPL-Lizenz verfügbar [31].

[1] http://topbraid.org/spin/api/
[2] http://www.topquadrant.com/

Kapitel 8

Zusammenfassung

In dieser Arbeit wurde gezeigt, wie ein regelbasierter Ansatz zur Informationsextraktion aus gescannten Rechnungen eingesetzt werden kann. Das vorgenommene Ziel, Intrastat-Meldungen automatisiert zu erstellen, wurde weitgehend erreicht, sodass die Grundvoraussetzungen für einen Betrieb in der Praxis geschaffen wurden.

Aufgrund des hohen Aufwands große Mengen von Rechnungen zu Scannen und der potentiellen Anfälligkeit für OCR-Fehler, sollte die Dokumentenverarbeitung prinzipiell nur dann in Betracht gezogen werden, wenn sich die benötigten Daten nicht in digitaler Form organisieren lassen. Ein scheinbar triviales Problem, das aber die Effizienz des gesamten Systems beeinträchtigt, können geklammerte Dokumente darstellen. Weiters muss sichergestellt werden, dass der Kontrollaufwand durch den Benutzer auf ein Minimum beschränkt wird und die Überprüfung der Daten zügig erfolgen kann. Neben den technischen Verbesserungsmöglichkeiten, liegt im Usability-Bereich noch viel Potential. Durch die farbliche Darstellung der Bewertung können die einzelnen Datenfelder, die einer Bearbeitung bedürfen, schnell gefunden werden. Die Überprüfung vieler Daten gestaltet sich allerdings noch mühsam. Indem der entsprechende Bereich im Dokument hervorgehoben wird, wurde die Überprüfung bereits erleichtert, allerdings muss das Auge ständig zwischen linkem und rechtem Teil der Anwendung hin und her springen. Das Interaktionskonzept Bedarf daher noch einer Überarbeitung. Das zugrundeliegende System lässt sich jedoch für beliebige Anwendungsfälle, in denen strukturierte Dokumente zu verarbeiten sind, adaptieren. Dazu ist im Wesentlichen nicht mehr nötig, als die tabellarische Übersicht des GUIs anzupassen und eventuell notwendige Daten zur Klassifizierung und Verifizierung bereitzustellen.

Neben den technischen Optimierungsmöglichkeiten sind Weiterentwicklungen in verschiedenen Bereichen denkbar. Ein großer offener Punkt, der bisher nicht behandelt wurde, betrifft die automatische Erkennung von Strukturen in unbekannten Dokumenten, sodass das manuelle Verfassen von Regeln

erleichtert und in weiterer Folge automatisiert wird.

Der stetig wachsende Datenbestand könnte in Zukunft weiters genutzt werden, um die erfassten Klassifizierungen semantisch zu analysieren und, darauf aufbauend, ein Vorschlagsystem zu entwickeln. Auf diese Weise könnten neue Artikel automatisch zu Tarifnummern zugeordnet werden, um den Bearbeitungsaufwand weiter zu reduzieren.

Quellenverzeichnis

Literatur

[1] R.L. Ackoff. „From Data to Wisdom“. In: *Journal of Applied Systems Analysis* 16.1 (1989), S. 3–9.

[2] E. Appiani u. a. „Automatic document classification and indexing in high-volume applications“. In: *International Journal on Document Analysis and Recognition* 4.2 (2001), S. 69–83.

[3] Statistik Austria. *Außenhandel: Meldeerleichterungen für zahlreiche Unternehmen ab Berichtsjahr 2010*. 2010. URL: http://www.statistik.at/web_de/static/pm_meldeerleichterungen_fuer_zahlreiche_unternehmen_ab_berichtsjahr_2010_043102.pdf.

[4] Statistik Austria. *Binnenhandelsstatistik – Anleitung zur Abgabe der INTRASTAT-Meldungen*. Jan. 2012. URL: http://www.statistik.at/web_de/static/binnenhandelsstatistikbranleitung_zur_abgabe_der_intrastat-meldungen_020485.pdf.

[5] Statistik Austria. *Binnenhandelsstatistik – EDI – Anleitung für die österreichische Version von INSTAT SUBSET von CUSDEC D.97B*. Dez. 2012. URL: http://www.statistik.at/web_de/static/binnenhandelsstatistik_edi-anleitung_fuer_die_oesterreichische_version_von_020487.pdf.

[6] Statistik Austria. *Binnenhandelsstatistik – Warenverzeichnis 2013*. 2012. URL: http://www.statistik.at/web_de/static/aussenhandel_warenverzeichnis_2013_kap.01-97_gueltig_ab_1.1.2013_069117.pdf.

[7] Statistik Austria. *Standard-Dokumentation Metainformationen (Definitionen, Erläuterungen, Methoden, Qualität) zu den Außenhandelsstatistiken*. Jan. 2011. URL: http://www.statistik.at/wcmsprod/groups/gd/documents/stddok/001650.pdf.

[8] Evgeniy Bart und Prateek Sarkar. „Information extraction by finding repeated structure“. In: *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*. DAS '10. Boston, Massachusetts: ACM, 2010, S. 175–182.

[9] Dave Beckett und Brian McBride. *RDF/XML Syntax Specification (Revised)*. Techn. Ber. World Wide Web Consortium (W3C), Feb. 2004. URL: http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/.

[10] David Beckett u. a. *Terse RDF Triple Language*. Techn. Ber. World Wide Web Consortium (W3C), Juli 2012. URL: http://www.w3.org/TR/2012/WD-turtle-20120710/.

[11] Paul V. Biron und Ashok Malhotra. *XML Schema Part 2: Datatypes Second Edition*. Techn. Ber. World Wide Web Consortium (W3C), Okt. 2004. URL: http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/.

[12] Dan Brickley, R.V. Guha und Brian McBride. *RDF Vocabulary Description Language 1.0: RDF Schema*. Techn. Ber. World Wide Web Consortium (W3C), Feb. 2004. URL: http://www.w3.org/TR/2004/REC-rdf-schema-20040210/.

[13] F Cesarini u. a. „Analysis and understanding of multi-class invoices". In: *International Journal on Document Analysis and Recognition* 6.2 (2003), S. 102–114.

[14] N. Chen und D. Blostein. „A survey of document image classification: problem statement, classifier architecture and performance evaluation". In: *International Journal on Document Analysis and Recognition* 10.1 (2007), S. 1–16.

[15] Florian Deckert u. a. „Table Content Understanding in SmartFIX". In: *Proceedings of the 2011 International Conference on Document Analysis and Recognition*. ICDAR '11. Washington, DC, USA: IEEE Computer Society, 2011, S. 488–492.

[16] Andreas R. Dengel. „Making Documents Work: Challenges for Document Understanding". In: *Proceedings of the Seventh International Conference on Document Analysis and Recognition – Volume 2*. ICDAR '03. Washington, DC, USA: IEEE Computer Society, 2003, S. 1026–1035.

[17] *European Central Bank: Daily nominal effective exchange rate of the euro*. URL: http://www.ecb.int/stats/exchange/effective/html/index.en.html.

[18] *Frequently Asked Questions About the Java HotSpot VM*. URL: http://www.oracle.com/technetwork/java/hotspotfaq-138619.html#gc_heap_32bit.

[19] Joseph C. Giarratano und Gary Riley. *Expert Systems: Principles and Programming*. 4. Aufl. Thomas Course Technology, 2005.

[20] Asunción Gómez-Pérez, Mariano Fernández-López und Oscar Corcho. *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. Springer, 2004.

[21] Thomas R Gruber. „A translation approach to portable ontology specifications“. In: *Knowledge Acquisition* 5.2 (1993). Hrsg. von Thomas R Gruber, S. 199–220.

[22] H Hamza, Y Bela Id und A Bela Id. „Case-Based Reasoning for Invoice Analysis and Recognition“. In: *Case-Based Reasoning Research and Development* (2007), S. 404–418.

[23] Steve Harris und Andy Seaborne. *SPARQL 1.1 Query Language*. Techn. Ber. World Wide Web Consortium (W3C), Nov. 2012. URL: http://www.w3.org/TR/2012/PR-sparql11-query-20121108/.

[24] Ian Horrocks u. a. *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. Techn. Ber. World Wide Web Consortium (W3C), Mai 2004. URL: http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/.

[25] Peter Jackson. *Introduction to Expert Systems*. 3. Aufl. Addison-Wesley, 2004.

[26] B. Klein und A.R. Dengel. „Problem-adaptable document analysis and understanding for high-volume applications“. In: *International Journal on Document Analysis and Recognition* 6.3 (2003), S. 167–180.

[27] Stefan Klink und Thomas Kieninger. „Rule-based Document Structure Understanding with a Fuzzy Combination of Layout and Textual Features“. In: *International Journal of Document Analysis and Recognition* 4.1 (2001), S. 18–26.

[28] Graham Klyne und Jeremy J. Caroll. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. Techn. Ber. World Wide Web Consortium (W3C), Feb. 2004. URL: http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/.

[29] Holger Knublauch. *SPIN – Modeling Vocabulary*. Techn. Ber. World Wide Web Consortium (W3C), Feb. 2011. URL: http://www.w3.org/Submission/2011/SUBM-spin-modeling-20110222/.

[30] Holger Knublauch. *SPIN – SPARQL Syntax*. Techn. Ber. World Wide Web Consortium (W3C), Feb. 2011. URL: http://www.w3.org/Submission/2011/SUBM-spin-sparql-20110222/.

[31] Holger Knublauch. *The TopBraid SPIN API 1.2.0*. Techn. Ber. TopQuadrant, 2011. URL: http://topbraid.org/spin/api/1.2.0/index.html.

[32] Europäische Kommission. *VAT Information Exchange System – FAQ*. URL: http://ec.europa.eu/taxation_customs/vies/faq.html.

[33] V. I. Levenshtein. „Binary Codes Capable of Correcting Deletions, Insertions and Reversals“. In: *Soviet Physics Doklady* 10 (Feb. 1966), S. 707.

[34] Frank Manola und Eric Miller. *RDF Primer*. Techn. Ber. World Wide Web Consortium (W3C), Feb. 2004. URL: http://www.w3.org/TR/2004/REC-rdf-primer-20040210/.

[35] Deborah L. McGuinness und Frank van Harmelen. *OWL Web Ontology Language - Overview*. Techn. Ber. World Wide Web Consortium (W3C), Feb. 2004. URL: http://www.w3.org/TR/2004/REC-owl-features-20040210/.

[36] X. Peng u. a. „Information Extraction from Historical Semi-Structured Handwritten Documents“. In: *Family History Technology Workshop 2012*. Salt Lake City, Utah, 2012, S. 20–24.

[37] Jennifer Rowley. „The wisdom hierarchy: representations of the DIKW hierarchy“. In: *Journal of Information Science* 33.2 (2007), S. 163–180.

[38] Faisal Shafait und Ray Smith. „Table detection in heterogeneous documents“. In: *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*. DAS '10. Boston, Massachusetts: ACM, 2010, S. 65–72.

[39] Ray Smith. „An Overview of the Tesseract OCR Engine“. In: *Proceedings of the Ninth International Conference on Document Analysis and Recognition – Volume 02*. ICDAR '07. Washington, DC, USA: IEEE Computer Society, 2007, S. 629–633.

[40] Raymond W. Smith. „Hybrid Page Layout Analysis via Tab-Stop Detection“. In: *Proceedings of the 2009 10th International Conference on Document Analysis and Recognition*. ICDAR '09. Washington, DC, USA: IEEE Computer Society, 2009, S. 241–245.

[41] R. Sulaiman, M.F.M. Amran und N.A.A. Majid. „A Study on Information Extraction Method of Engineering Drawing Tables“. In: *International Journal of Computer Applications* 50.16 (2012), S. 43–47.

[42] Europäische Union. *Durchführungsverordnung (EU) Nr. 927/2012 der Kommission vom 9. Oktober 2012 zur Änderung von Anhang I der Verordnung (EWG) Nr. 2658/87 des Rates über die zolltarifliche und statistische Nomenklatur sowie den Gemeinsamen Zolltarif*. Amtsblatt der Europäischen Union. URL: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2012:304:FULL:DE:PDF.

[43] Europäische Union. *Richtlinie 2006/112/EG des Rates vom 28. November 2006 über das gemeinsame Mehrwertsteuersystem*. Amtsblatt der Europäischen Union. URL: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2006:347:0001:0118:de:PDF.

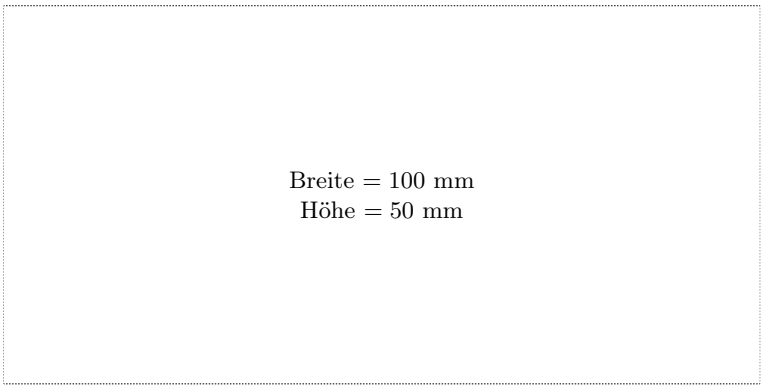
[44] Europäische Union. *Verordnung (EG) Nr. 222/2009 des Europäischen Parlaments und des Rates vom 11. März 2009 zur Änderung der Verordnung (EG) Nr. 638/2004 über die Gemeinschaftsstatistiken des Warenverkehrs zwischen Mitgliedsstaaten*. Amtsblatt der Europäischen Union. URL: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2009:087:0160:0163:DE:PDF.

[45] Europäische Union. *Verordnung (EG) Nr. 638/2004 des Europäischen Parlaments und des Rates vom 31. März 2004 über die Gemeinschaftsstatistiken des Warenverkehrs zwischen Mitgliedsstaaten und zur Aufhebung der Verordnung (EWG) Nr. 3330/91 des Rates*. Amtsblatt der Europäischen Union. URL: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2004:102:0001:0008:DE:PDF.

[46] W3C. *Ontologies*. URL: http://www.w3.org/standards/semanticweb/ontology.

Messbox zur Druckkontrolle

— Druckgröße kontrollieren! —



— Diese Seite nach dem Druck entfernen! —