universität wien

MASTERARBEIT

Titel der Masterarbeit

Automatische Wiedererkennung von Webseiten auf Basis von "Fingerabdrücken"

Verfasser

Bakk.rer.soc.oec.
Wolfgang Jandl

angestrebter akademischer Grad

Diplom-Ingenieur

Wien, 2009

Studienkennzahl lt. Studienblatt:	066 926
Studienrichtung lt. Studienblatt:	Wirtschaftsinformatik
Betreuer:	ao. Univ.-Prof. Dr. Werner Retschitzegger

Wolfgang Jandl
Stättermayergasse 32/10
1150 Wien

Hiermit erkläre ich an Eides statt, dass ich die von mir vorgelegte Arbeit selbstständig verfasst habe, dass ich die verwendeten Quellen, Internet-Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, den 15. November 2009

Wolfgang Jandl

Inhaltsverzeichnis

Danksagung **3**

1 Motivation **5**

2 Stand der Technik **9**
2.1 Information Retrieval vs. Information Extraction 9
2.2 Teilbereiche der Information Extraction 12
2.2.1 Navigation 12
2.2.2 Extraktion 15

3 Inhalts- und Seitenanalyse **17**
3.1 Einführung 17
3.2 Analyse der Quell-Webseiten 19
3.2.1 URLs 21
3.2.2 HTML-Struktur 22

4 Erstellung und Vergleich von Fingerabdrücken **31**
4.1 Erstellung von Fingerabdrücken 31
4.1.1 URLs 31
4.1.2 Formulare 32
4.1.3 HTML-Tags mit ID-Attribut 36
4.1.4 HTML-Teilbaum 38
4.2 Vergleich von Fingerabdrücken 40
4.2.1 Punktesystem 40
4.2.2 URLs 42
4.2.3 Formulare 44
4.2.4 HTML-Tags mit ID-Attribut 45
4.2.5 HTML-Teilbaum 46

5 Prototypische Implementierung **49**
5.1 Anwenwendungsvorgaben 49
5.2 Essenzielle Klassen 51
5.3 Eingesetzte Fremdentwicklungen 55

6 Evaluierung **57**

6.1 Testfälle 57
6.2 Testergebnisse 60

7 Verbesserungspotential **67**

Abbildungsverzeichnis **71**

Tabellenverzeichnis **73**

Listings **75**

Quellenverzeichnis **79**

A Klassendiagramm **81**

B Installationsanleitung **83**

C Kurzfassung **85**

D Abstract **87**

E Lebenslauf **89**

Danksagung

Diese Arbeit stellt den Abschluss meines Studiums der Wirtschaftsinformatik an der Universität Wien dar. Hiermit möchte ich mich bei meinen Eltern bedanken, die mir dieses Studium ermöglicht haben.

Darüber hinaus bedanke ich mich bei meinem Betreuer ao. Univ.-Prof. Dr. Werner Retschitzegger der mir mit seinen vielen Verbesserungsvorschlägen, Tipps und Hinweisen sehr bei der Verbesserung der Qualität dieser Arbeit geholfen hat.

Weiters möchte ich mich bei der Lixto Software GmbH bedanken, die mir die Möglichkeit gegeben hat schon vor dem erfolgten Studienabschluss in einem professionellen IT-Unternehmen zu arbeiten und praktische Erfahrungen zu sammeln, die zu vermitteln für eine Universität ja bekanntlich besonders schwierig sind. Die Idee zu dieser Diplomarbeit ist aus meinen Bedürfnissen als Wrapper-Entwickler entstanden und wurde insbesondere vom Leiter der Entwicklungsabteilung des Unternehmens, Mag. Dr. Robert Baumgartner, unterstützt. Ihm gilt mein besonderer Dank, denn in seinen Lehrveranstaltungen an der TU Wien wurde ich an das Thema Webdaten-Extraktion herangeführt.

Kapitel 1

Motivation

Die automatische Extraktion von Informationen aus dem Internet bietet den Nutzern solcher Dienste viele Vorteile [1], wie zum Beispiel die Möglichkeit aus den semistrukturierten, verteilten Daten wie sie im Web vorhanden sind strukturierte und konzentrierte Daten zu erzeugen, indem sie zum Beispiel in Datenbanksystemen gespeichert werden. Im Allgemeinen spricht man im Zusammenhang mit computergesteuerter Web-Navigation und Informations-Extraktion von *web-crawlern* [2] oder auch *Wrappern*. Solche Applikationen, einen guten Überblick über dieses Thema geben [3], [4] und [5], haben die Aufgabe auf ganz bestimmte Art und Weise durch zuvor definierte Webseiten zu navigieren um ganz bestimmte Informationen auszulesen und von der ursprünglich beinhaltenden Webseite unabhängig verfügbar zu machen. Unternehmen die solche Services in Anspruch nehmen, haben häufig den Wunsch sich über den online-Markt ihrer Branche zu informieren, da dieser ein höchst transparenter und für die Kunden sehr leicht einsehbarer Markt ist. Ein weiteres Einsatzszenario für Technologien zur automatisierten Datenextraktion aus dem Internet sind Applikationen zur Metasuche, bei denen Webseitenbetreiber es ihren Kunden ermöglichen möchten mehrere Anbieter vergleichbarer Services oder Produkte gleichzeitig abzufragen um Angebote besser, leichter und effizienter vergleichen zu können. Häufig angewendet werden diese Prinzipien bei der Suche nach Hotelzimmern über mehrere Quellen, wie dies zum Beispiel auf den Seiten der Österreich-Werbung [1] der Fall ist.

Um solche Szenarien umsetzen zu können muss die Extraktionsapplikation dazu in der Lage sein verschiedene Webseiten auf jeweils spezifische Art und Weise zu behandeln. Will sich beispielsweise ein Anbieter von Hotelzimmern über die Preisgestaltung seiner Konkurrenten informieren, so kann er die Webseiten seiner Konkurrenten mit für ihn interessanten Parametern für Reiseziel und Aufenthaltsdauer automatisiert abfragen lassen. Dabei hat das System verschiedene Aufgaben; zuerst muss ein HTML-Formular mit den gewünschten Parametern befüllt und abgeschickt werden bevor die passenden Suchergebnisse angezeigt werden können. Hat man den Wunsch spezielle Informationen zu extrahieren muss jedes angebotene Hotel einzeln aufgerufen werden um sich beispielsweise die Adresse der Unterkunft anzeigen lassen zu können. In diesem Bei-

[1] http://www.austria.info/at/hotel-ferienhaus-buchen/

spielfall besteht die Navigation also aus drei Schritten und jede der drei angezeigten Webseiten der behandelten Domäne verlangt spezifische Behandlung. Die Autoren von [6] beschreiben diese Vorgehensweise als *Deep Web Navigation*.

Um solch komplexe Navigationen von einem automatischen System durchführen zu lassen sollte es für das System von Vorteil sein zu wissen welche Aufgaben und Vorgänge für einen bestimmten Typ von Webseite vorgesehen sind. Durch die Fähigkeit der Applikation zu wissen welche Aufgaben sie im Zusammenhang mit der zu diesem Zeitpunkt vorliegenden Homepage hat ergibt sich im Navigations-Workflow ein Mehr an Sicherheit. Denn ist sich ein Wrapper *"bewusst"* auf welcher Art von Seite er sich gerade befindet kann er seinen Verhalten dementsprechend anpassen und jederzeit die für diesen Seitentypus vorgesehenen Mechanismen zur Navigation oder Extraktion aktivieren.

Es kann vorkommen, dass eine automatische Navigation durch gewisse Umstände in ihrem Ablauf behindert oder gestört wird. Dies können zum Beispiel Ladefehler und Verbindungsabbrüche oder fehlerhafte Verlinkungen und dergleichen sein, wodurch eine Navigation aus ihrem vorgesehenen Ablauf gezwungen wird. Tritt nun also während der Abarbeitung einer Folge von Webseitenaufrufen eine Störung auf sodass sich der Wrapper auf einer Seite wiederfindet die er zu diesem Zeitpunkt "nicht erwartet" hatte, soll ein *Fingerabdruck* der angezeigten Webseite erzeugt werden. Zuvor wurde bei der Erstellung der Navigations- und Extraktionsschritte vom Entwickler von jeder zu bearbeitenden Seitenart ein Fingerabdruck erstellt und benannt; durch Vergleich des aktuellen Fingerabdrucks der offenbar nicht der korrekten Reihenfolge entsprechenden Seite mit den vom Benutzer zuvor angelegten Fingerabdrücken kann festgestellt werden ob die aktuelle Seite grundsätzlich einem der vorgegebenen Muster entspricht und somit einem bestimmten Teilbereich der Navigation oder Extraktion zugewiesen werden kann. Dadurch wird eine korrekte Fortführung der Navigation ermöglicht und die Abarbeitung der Aufgaben des Wrappers robuster gegenüber Seitenfehlern gemacht. Kann jedoch keine vergleichbare zu bearbeitende Seite gefunden werden wird dies dem Wrapper-Ersteller auf geeignete Weise zur Kenntnis gebracht.

Für den Menschen ist es ein leichtes die Aufgabe einer Webseite auf den ersten Blick zu erkennen. Ein Computer hingegen kann zwar den Inhalt einer Webseite einlesen und mittels geeigneter Programme zur Darstellung bringen, den Sinn und Zweck der dargestellten Komponenten zu bestimmen ist im Allgemeinen jedoch unmöglich. Die vorliegende Arbeit hat nicht zum Ziel einem Rechner die Fähigkeit zu geben die Aufgaben von Internetseiten selbständig bestimmen zu können. Viel mehr geht es darum es einem Computer zu ermöglichen Webseiten auf Basis ihrer Inhalte miteinander zu vergleichen und bestimmen zu können ob zwei Seiten vom selben Typ sind oder nicht. Dabei soll es nicht um zwei Seiten gehen die in keiner Beziehung zueinander stehen, sondern immer nur um einzelne *Unter-Webseiten* einer bestimmten Domäne. Es soll also beispielsweise erkannt werden können, dass die beiden Webseiten der Abbildungen 1.1 und 1.2 vom selben Webseitentyp sind (Auflistung von Suchergebnissen), während 1.3 andere Aufgaben zu erfüllen hat (Details zu einem Angebot anzeigen). Wie man sieht sind alle drei dargestellten Webseiten so genannte *Unter-Webseiten* oder *Sub-*

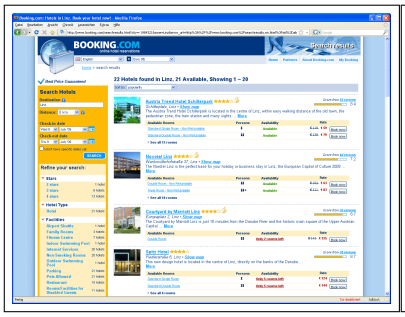


Abbildung 1.1:
Beispielseite 1

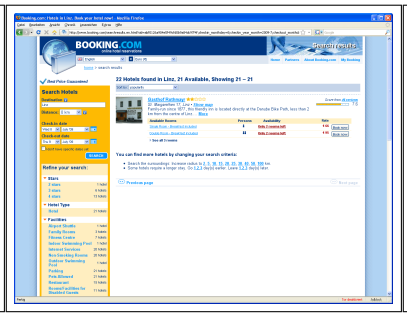


Abbildung 1.2:
Beispielseite 2

Abbildung 1.3:
Beispielseite 3

seiten der Domäne http://www.booking.com, denn ein Wrapper ist im Allgemeinen immer nur für eine bestimmte Domäne entworfen und muss ausschließlich Webseiten der selben Domäne bearbeiten. Realisiert werden soll die Wiedererkennung über den erwähnten sog. "*Fingerabdruck*" der Webseite, der sich aus verschiedenen in der Webseite verwendeten Konstrukten und Inhalten ergibt und mit Fingerabdrücken anderer Webseiten verglichen wird.

Das ganze System ist im Bereich der mittels Wrappern automatisierten Extraktion von Daten aus dem Internet angesiedelt, bei der im Allgemeinen so vorgegangen wird, dass ein Wrapper immer nur eine bestimmte Website, jedoch unterschiedlichste Subseiten dieser Domäne zu bearbeiten hat. Zur Nutzung der Fingerabdrücke und deren Vergleichsmöglichkeiten soll der Ersteller des Wrappers von jeder zu bearbeitenden Seite zur Entwicklungszeit einen Fingerabdruck erzeugen lassen um ihn von der Applikation im "Pool der bekannten Webseiten" speichern zu lassen. Der Wrapper selbst soll dann wie bereits beschrieben dazu in der Lage sein die zu diesem Zeitpunkt angezeigte Webseite einer der vom Ersteller des crawlers als bekannt definierten Seiten zuzuweisen. Es soll der Computer also selbständig erkennen können, dass die Seiten der Abbildungen 1.1 und 1.2 sehr ähnliche Inhalte und auch die selbe Aufgabe haben und dadurch als gleich zu betrachten sind, während 1.3 einer anderen Kategorie entspricht.

Die Vorteile die sich daraus ergeben sollen dazu beitragen die Techniken und Technologien zur automatisierten und strukturierten Extraktion von Daten aus dem WWW zu unterstützen und zu verbessern sowie auf die Herausforderungen in diesem Gebiet zu reagieren.

Der Aufbau der vorliegenden Arbeit gestaltet sich wie folgt:

Zu Beginn wird in Kapitel 2 auf aktuelle Ansätze und Enwicklungen auf dem Gebiet der automatischen Navigation und Datenextraktion aus dem WWW eingegangen.

Danach werden in Kapitel 3 jene Bestandteile von Webseiten beschrieben, die prinzipiell zur Erstellung von Fingerabdrücken berücksichtigt werden könnten. Dabei wird ausgeführt warum bestimmte Inhalte in einen Fingerabdruck mit einfließen oder eben keine Berücksichtigung finden.

Daran anschließend wird festgehalten wie die Erstellung eines Fingerabdrucks tatsäch-

lich umgesetzt wurde und wie die einzelnen Webseiten-Komponenten darin mit einfließen. Im selben Abschnitt 4 findet sich außerdem die Beschreibung der Vorgehensweise beim Vergleich zweier Fingerabdrücke miteinander.

Das Kapitel 5 gibt eine Übersicht über die Umsetzung der Applikation auf Programmiersprachenebene und streicht die wichtigsten Komponenten, Strukturen und Techniken hervor bzw. illustriert die Umsetzung anhand von ausgewählten Diagrammen und der Beschreibung der wichtigsten Programmbestandteile. Zur Erstellung der Applikation wurde die objektorientierte Programmiersprache *Java* verwendet.

Im Abschnitt 6 werden die durchgeführten Tests beschrieben und deren Ergebnisse diskutiert, bevor in Kapitel 7 Verbesserungsvorschläge und Änderungen bzw. Erweiterungen der Applikation beschrieben werden.

Kapitel 2

Stand der Technik

Die in dieser Arbeit vorgestellte Applikation ist im Feld der automatisierten Datenextraktion aus dem Internet angesiedelt. Der vorliegende Abschnitt beschreibt die grundlegenden Konzepte in diesem Gebiet und damit in welchem Kontext die Arbeit zu sehen ist. Es wird auf bestehende Ansätze auf diesem Gebiet eingegangen sowie festgestellt, wie sich die in dieser Arbeit beschriebene Applikation in die Entwicklungen auf diesem Gebiet einfügt. Zuvor werden einige grundlegende Begriffe geklärt und es wird kurz auf die Historie des Forschungsgebiets eingegangen.

2.1 Information Retrieval vs. Information Extraction

Das WWW kann als eine riesige "Datenbasis" angesehen werden in der Informationen teilstrukturiert zur Verfügung stehen. Die im Internet verfügbare Datenmenge ist meist semistrukturiert, da sie im Vergleich zu unstrukturierten Fließtexten mittels der Auszeichnungssprache HTML über die Daten beschreibende Strukturen verfügt (beispielsweise Tabellen oder Listen). Diese Möglichkeiten sind jedoch nicht mit strukturierten Informationen in Datenbanken vergleichbar wo die Daten nach streng definierten Regeln in exakten Strukturen hinterlegt sind. Die Hauptaufgabe der automatischen Extraktion von Daten aus dem Internet besteht darin die semistrukturierten Inhalte des WWW in strukturierte Daten umzuwandeln um diese für bestimmte Anwendungen einfacher zugänglich zu machen.

Auf dem Gebiet der Informationssammlung und -strukturierung gibt es zwei voneinander zu unterscheidende Begriffe; einerseits spricht man von *Information Retrieval (IR)*, andererseits von *Information Extraction (IE)*. Die historische Veröffentlichung von V. Bush [7] in der er sein "memex" System beschreibt ist aus heutiger Sicht die wohl erste Beschreibung eines Systems das Informationen aus verschiedenen Quellen anhand von so genannten "trails" miteinander zu verknüpfen in der Lage sein sollte und eine der Grundlagen des Information Retrieval darstellt. Mit memex ist auch das heute bestehende Internet vergleichbar das ja auch aus Dokumenten besteht die auf andere Dokumente verweisen. Die aktuell meistgenutzte Suchmaschine des WWW - Google [8]

- nutzt solche Verknüpfungen zwischen Dokumenten um Ranglisten von Suchergebnissen zu erstellen und führt somit die Denkansätze von V. Bush fort. Darüber hinaus sind Internet-Suchmaschinen ganz allgemein darauf ausgerichtet bestimmte Worte (Suchbegriffe) in im WWW veröffentlichten Dokumenten zu finden und die gefundenen Inhalte als Ergebnis der Suche aufzulisten. Der Begriff Information Retrieval wurde bereits 1951 erstmals durch Calvin N. Mooers verwendet [9] und erlangte durch die Arbeiten von Gerald Salton in den 1980er Jahren besondere Bedeutung [10].

Information Retrieval bezieht sich also darauf Informationen in verschiedenen Quellen ausfindig und dem Nutzer zugänglich zu machen. Dabei werden die gesuchten Informationen nicht aus den gefundenen Dokumenten extrahiert, sondern die gesamten Dokumente als Suchergebnis geliefert. Eine Herausforderung auf diesem Gebiet ist die Fähigkeit des Suchsystems zu erkennen welche gefundenen Quellen von größerer und welche von geringerer Relevanz sind, um dem Nutzer so Ergebnisse mit für ihn wertvolleren Informationen vorrangig anzeigen zu können. Mittlerweile existieren auch Suchmaschinen im Internet, die Suchergebnisse in bestimmte Gruppen einteilen was auch auf Technologien des Information Retrieval zurückzuführen ist. Beispielsweise präsentiert die Suchmaschine cuil [1] zum Suchbegriff "Niki Lauda" nicht nur Details zu seiner Person, sondern auch zu Menschen aus seinem Umfeld die in mit ihm vergleichbare Kategorien einzuteilen sind (zum Beispiel Formel 1 Rennfahrer, Formel 1 Weltmeister, usw.).

Im Gegensatz dazu befasst sich Information Extraction damit gewisse Inhalte aus Dokumenten zu extrahieren und diese weiterzuverarbeiten. Von Relevanz in diesem Zusammenhang sind also nicht die Dokumente an sich, sondern ganz bestimmte Inhalte davon. Die gewonnenen Fakten werden durch ihre Aggregierung und Zusammenführung mit vergleichbaren Daten in strukturierte Informationen umgewandelt wodurch sie zu einem größeren Ganzen werden aus dem wiederum weitere Informationen abgeleitet werden können. Ein Beispiel dafür ist die strukturierte Extraktion von Preisen für bestimmte Güter, beispielsweise Hotelzimmer, die allein für sich genommen wenig aussagekräftig sind. Ist man jedoch dazu in der Lage Preisinformationen zu ein und demselben Hotel aus verschiedenen Quellen zusammenzuführen so erschließt sich einem die Möglichkeit sich einen Überblick über die Marktpreise zu verschaffen, da die Angebote miteinander vergleichbar werden. Durch die Miteinbeziehung verschiedener weiterer Fakten zu einem Hotel wie der inkludierten Verpflegung oder anderen Sonderleistungen lässt sich ein umfassendes Marktbild erstellen und es wird dem Nutzer dieser Angebote ermöglicht sich einen höchst transparenten und übersichtlichen Markt zu schaffen.

Um einen solchen Marktüberblick umzusetzen reicht es also nicht aus wie bei Suchmaschinen üblich Webseiten zu suchen die bestimmte Begriffe enthalten und diese als Ergebnis zu liefern, sondern man muss dazu in der Lage sein mit den Webseiten zu interagieren, um gewisse Informationen überhaupt erst zur Anzeige bringen zu können. Dazu zählen die vielen Angebote im WWW die sich an einen beschränkten Personenkreis richten und beispielsweise nur über Identifikation mittels Benutzernamen und

[1] http://www.cuil.com

Passwort zugänglich sind. Weiters werden jene Informationen dazu gezählt, die eigentlich in Datenbanken verborgen liegen und über dynamische Generierung von Webseiten temporär im Internet veröffentlicht werden. Lässt sich eine solche Seite nicht mehr anzeigen - zumeist wegen abgelaufener Sessions - sind auch die darin enthaltenen Informationen nicht mehr zugänglich und müssen neu generiert werden bevor sie wieder sichtbar gemacht werden können. Diese "versteckten" Informationen werden als "Deep Web" bezeichnet, weil sie hinter virtuellen "Mauern" verborgen sind, die nur von dazu berechtigten Personen mit geeigneten Mitteln überwunden werden können. Diesen Begriff des Deep Web hat Michael K. Bergmann 2001 in einem Whitepaper [11] geprägt. Dabei werden Suchmaschinen mit Fischern verglichen die ihre Schleppnetze über den Ozean ziehen und so in der Lage sind große Mengen von Meereslebewesen zu fangen. Verglichen jedoch mit dem Leben das in tiefer liegenden Meereszonen existiert sind die von der Oberfläche gesammelten Lebewesen nur ein kleiner Teil des Ganzen und die Fischer sind sich gar nicht bewusst was sie noch an Land ziehen könnten würden sie dazu in der Lage sein tiefere Meeresregionen zu befischen.

Bergmann versucht in seiner Arbeit auch einen Überblick über die im Internet vorhandenen Datenmengen getrennt nach Deep Web und "oberflächlichem Web" zu geben und hat Zahlen veröffentlicht die für sich sprechen. So sollen die im Deep Web vorhandenen Informationen in ihrem Umfang zwischen 400 und 550 mal größer sein als alle Daten des üblicherweise als WWW bezeichneten Teilbereichs des Internets. Die Datenmenge des Deep Webs wurde für 2001 mit einem Volumen von ca. 7.500 Terabyte angegeben, während das oberflächliche Web aus nur etwa neunzehn Terabyte an Daten bestand. Auf seiner Homepage [2] hat Bergmann Anfang 2007 seine Angaben jedoch relativiert und sprach davon dass der Deep Web Content etwa 100 mal so groß sei wie die Daten im oberflächlichen Web ohne dafür jedoch neue Zahlen zu nennen. Chang et. al. [12] haben im Jahr 2004 ebenfalls Untersuchungen zum Deep Web angestellt und sprechen von über 300000 im WWW verfügbaren Datenbanken und über 1,2 Millionen Schnittstellen um auf diese zuzugreifen, jedoch nicht von konkreten Datenmengen. Es ist jedoch auf jeden Fall davon auszugehen dass die Informationsfülle in den vergangenen Jahren - sowohl im Deep Web als auch im oberflächlichen WWW - weiter enorm gewachsen ist.

Diese vermutlich riesige Menge an Informationen des Deep Web zu erreichen stellt eine Herausforderung im Zusammenhang mit Information Extraction dar, denn es besteht nicht nur die Schwierigkeit ganz bestimmte Daten aus bestimmten Webseiten auszulesen, sondern man muss auch über diverse Zwischenseiten navigieren und mit den einzelnen Webseiten auf verschiedene Weisen interagieren um diejenigen Webseiten anzeigen lassen zu können, die relevante Informationen enthalten.

Erfolgreiche Datenextraktion aus dem Deep Web setzt sich also aus den zwei grundlegenden Komponenten *Navigation* und *Extraktion* zusammen. In den vergangenen Jahren wurden viele Arbeiten zum Thema Datenextraktion veröffentlicht, während das Feld der Navigation durch Webseiten relativ wenig Aufmerksamkeit erfuhr.

[2] http://www.mkbergman.com/343/the-murky-depths-of-the-deep-web/

2.2 Teilbereiche der Information Extraction

Anwendungen der Information Extraction unterscheiden sich nicht nur in den eingesetzten Technologien und Techniken, sondern auch in ihren Zielen. In [13] beschreibt L. Eikvil drei unterschiedliche Einsatzgebiete der IE mit ihren verschiedenen Aufgaben und Absichten. Diese sind der Einsatz in Freitext, teilstrukturierten und strukturierten Texten.

Bei der Extraktion aus unstrukturierten Texten zielt man darauf ab die wichtigsten Informationen des Inhalts zu finden und wiederzugeben, also eine Zusammenfassung der Inhalte zu erstellen, wie dies auch in [14] oder [15] beschrieben ist. Zur Extraktion von Inhalten aus Freitexten kommen hauptsächlich Kenntnisse aus dem Gebiet des Natural Language Processings (NLP) [16] zum Einsatz. Die Anwendungen auf diesem Gebiet haben sich in den vergangenen Jahren weit entwickelt, sie sind jedoch von den menschlichen Fähigkeiten zur Textanalyse und Interpretation noch weit entfernt.

Erkenntnisse des NLP kommen bei der Extraktion von Informationen aus semistrukturierten Texten nicht zum Zug. Semistrukturierte Dokumente enthalten neben Fließtext auch noch strukturelle Elemente die jedoch nicht notwendigerweise strikt vorgegeben und standardisiert sind, sondern variabel sein können. Dadurch dass der Fließtext in solchen Dokumenten von den strukturellen Inhalten aufgespalten und zerteilt wird können NLP Algorithmen in diesen Fällen nicht problemlos angewendet werden, da die üblichen Textstrukturen und Satzzeichen nicht mehr korrekt gegeben bzw. angeordnet sein müssen. Semistrukturierte Texte sind beispielsweise HTML oder XML-Dokumente.

Von strukturiertem Text spricht man hingegen wenn Textinhalte gemeint sind, die ausschließlich in exakt definierten Strukturen enthalten sind. So kann man jede zugängliche relationale Datenbank mit den gleichen Kommandos und Befehlen abfragen, während HTML-Dokumente mit ihren variablen Tag-Namen zur Inhaltsanalyse nicht immer gleich be- und verarbeitet werden können. Die Bearbeitung von strukturierten Texten stellt im Allgemeinen keine große Herausforderung dar.

Die vorliegende Arbeit befasst sich grundsätzlich mit der Analyse und Datenextraktion aus HTML-Dokumenten, die, wie erwähnt, zu den semistrukturierten Dokumenten zu zählen sind. Das Hauptaugenmerk liegt dabei jedoch nicht auf der Extraktion von für den Benutzer von Extraktionssystemen relevanten Inhalten, sondern nutzt die Mittel der Informationsextraktion zur Verbesserung der Navigation im Deep Web.

2.2.1 Navigation

Die notwendige Navigation über Webseiten und die Interaktion mit diesen um Daten aus dem Deep Web an das oberflächliche Web zu holen ist ein wesentlicher Bestandteil der automatisierten Datenextraktion aus dem Internet. Bisher wurde jedoch, verglichen mit Arbeiten zur Extraktion von Daten, erst wenig zu diesem Bereich veröffentlicht. Die Veröffentlichung von V. Anupam et. al. [17] beschreibt eine Anwendung namens "WebVCR" (VCR steht für "Video Cassette Recorder") die es ermöglicht so genannte

"Smart Bookmarks" einmal aufzuzeichnen und beliebig oft wieder abzuspielen. Die Anwendung orientiert sich in ihren Prinzipien und der Benutzung an einem Videorekorder, deshalb auch der Name WebVCR. Die Entwickler des WebVCR sehen ihre Applikation als Mittel zur Zeitersparnis für Menschen die gewisse Folgen von Webseiten auf die immer gleiche Weise behandeln möchten. Ein vorgeschlagenes Anwendungsszenario ist die Suche nach Flügen auf einer bestimmten Webseite nach den stets selben Parametern, wobei WebVCR dazu in der Lage ist vom Nutzer im Browser durchgeführte Aktionen wie Webseitenaufrufe, Klicks oder Eingaben von Daten in Textfelder über ein Java-Applet zu registrieren und aufzuzeichnen. Die in den "Smart Bookmarks" aufgezeichnete Folge von Aktionen und Navigationsschritten kann nach erfolgter Aufzeichnung "abgespielt" werden und führt den Benutzer zu der gewünschten Webseite und den Daten von Interesse ohne dass der Benutzer selbst jedesmal die dafür notwendigen Schritte durchführen muss. Die dabei verwendeten Bestandteile der einzelnen Webseiten werden über das *Document Object Model (DOM)* [18] angesteuert und anhand ihrer Inhalte identifiziert. Nachdem sich die Position zu verwendender Elemente im DOM der Webseite jederzeit ändern kann wird bei Links beispielsweise verglichen, ob neben der Position im DOM auch noch die im href Attribut enthaltene Information sowie der Textinhalt des Link-Tags übereinstimmt. Auf diese Weise versucht man die aufgezeichneten Navigationen robust gegenüber Änderungen in den verwendeten Webseiten zu machen.

Neben dieser Arbeit ist der in [6] beschriebene Ansatz anzuführen, der sich dezidiert auf *"Deep Web Navigation in Web Data Extraction"* bezieht und die Applikation *"Lixto Visual Developer"* beschreibt. Der Visual Developer der Lixto Software GmbH [3] mit Sitz in Wien ist eine Applikation die es dem Benutzer ermöglicht ähnlich wie im WebVCR Aktionen aufzuzeichnen um sie danach automatisiert ablaufen zu lassen. Dabei werden sowohl Methoden zur Umsetzung der Navigation als auch der Extraktion angeboten. Die aktuell größten Herausforderungen und neuesten Entwicklungen der Anwendung werden in [19] und [20] näher erläutert.

Der Lixto Visual Developer hat die Fähigkeit komplexe Navigationsstrukturen in einem Wrapper abzubilden und automatisch auszuführen. Zur Entwicklung von Navigationssequenzen stehen dem Benutzer alle Möglichkeiten offen die man auch bei der "normalen" Interaktion mit einem Webbrowser hat, also URL-Aufrufe, das Befüllen von Textboxen, Klicks auf Links, Buttons, Radioboxes usw. Diese Möglichkeiten eröffnen der Applikation den Zugriff auf Daten des Deep Webs die nur über die Nutzung von Formularen zugänglich sind. Wie die Applikation im Unternehmensumfeld eingesetzt werden kann und welche Vorteile ein Unternehmen daraus gewinnen kann wird in [1] beschrieben.

Die in der vorliegenden Arbeit beschriebene Applikation ist inspiriert von den Fähigkeiten des Lixto Visual Developers und einem der Beispiele aus [1] und soll dabei von Nutzen sein einer der Herausforderungen in diesem Zusammenhang entgegenzutreten. Wie von den Entwicklern von Lixto beschrieben ist der Visual Developer dazu in der Lage komplexe Navigationen über beliebige Webseiten durchzuführen. Der Visual De-

[3] http://www.lixto.com

veloper hat jedoch nicht die Fähigkeit "zu verstehen" welche Aktionen durchzuführen sind, wenn eine bestimmte Webseite angezeigt wird. Er kann nur seinem vom Wrapper-Entwickler vorgegebenen Pfad folgen und führt nacheinander alle vorgegebenen Aktionen aus. Je komplexer eine Navigationsstruktur aufgebaut ist und je mehr Webseiten darin miteinbezogen werden desto größer wird die Wahrscheinlichkeit, dass im Verlauf der Abarbeitung irgendeine Art von Fehler auftritt.

Dabei kann man zwei verschiedene Arten von Fehlern unterscheiden:

1. Fehler im Wrapper
2. Fehler in der zu bearbeitenden Webseite

Fehler die im Wrapper enthalten sind treten auf, wenn eine vom Inhalt und Ablauf her korrekte Seite vollständig geladen wurde, die auszuführenden Aktionen jedoch trotzdem nicht durchgeführt werden können. Das heißt, dass beispielsweise ein bestimmter Link nicht geklickt werden kann weil die Position des betreffenden Verweises im HTML-Code nicht korrekt im Wrapper abgebildet wurde. Dies würde dazu führen, dass die folgenden Navigationsschritte ebenfalls nicht vollständig abgearbeitet werden können was zumeist auch dazu führt, dass die gesuchten Daten nicht angezeigt und auch nicht extrahiert werden können. Bezüglich der Extraktion selbst kann es ebenfalls zu Fehlern bei der Erstellung des Wrappers kommen die Variabilitäten in der Darstellung miteinander vergleichbarer Inhalte nicht berücksichtigen und dadurch ebenfalls die Extraktion der gewünschten Daten blockieren. Diese Fehler sind jedoch auf mangelnde Qualität der Wrapperentwicklung zurückzuführen und deren Bereinigung bleibt dem Entwickler überlassen.

Die Navigationsschritte eines Wrappers sind ein starrer Pfad von aneinandergereihten Aktionen die teilweise stark voneinander abhängen. Man betrachte beispielsweise das Szenario das zur Entwicklung der in dieser Arbeit vorgestellten Applikation herangezogen wurde. Die relevante Navigation besteht aus drei verschiedenen Schritten:

1. Befüllen eines Formulars mit den gewünschten Suchparametern,
2. Auswahl eines geeigneten Hotelzimmers von einer Seite die verfügbare Hotels mit Angeboten auflistet, und
3. Extraktion relevanter Daten aus einer Seite, die Details zum Zimmer (und evtl. auch Hotel) enthält.

Nachdem die Startseite vollständig geladen und das dort enthaltene Formular mit allen gewünschten Parametern befüllt ist wird auf einen Button geklickt der die eingetragenen Werte an den Seitenbetreiber sendet und diesen so dazu bringt zu den Eingaben passende Informationen aus seiner Datenbank temporär zu veröffentlichen. Sind die Parameter erfolgreich versendet, so muss gewartet werden bis die Folgeseite vollständig geladen ist bevor die nächsten Aktionen durchgeführt werden können. Im Normalfall sollte auf das Absenden des Formulars eine Webseite folgen, die einen Überblick über diverse Hotels und deren Angebote liefert. Um zu jedem Angebot genauere Daten extrahieren zu können ist es zumeist notwendig auf einen Link zu klicken der eine weitere

Seite aufruft die Verpflegung, Stornobedingungen und dergleichen enthält. Nachdem die Klick-Aktion durchgeführt wurde baut sich im Normalfall die anzuzeigende Seite mit den Angebotsdetails auf und die entsprechenden Daten können ausgelesen werden. Ist dies jedoch nicht der Fall und wird aufgrund falscher Verlinkungen oder Änderungen in der Verfügbarkeit eines Angebots nicht die eigentlich erwartete Webseite angezeigt sondern eine andere, so kann die dem letzten Klick folgende Extraktionsaktion nicht durchgeführt werden.

Genau hier setzt die Applikation der vorliegenden Arbeit an die das Ziel hat bei solchen Unregelmäßigkeiten in der Abfolge der angezeigten Webseiten die Funktionalität des Wrappers aufrecht zu erhalten. Gelingen soll dies, wie bereits in Kapitel 1 beschrieben, indem bestimmte Bestandteile der angezeigten Webseite extrahiert und analysiert, sowie mit den Bestandteilen von zur Entwicklungszeit definierten Webseiten verglichen werden. Dabei soll festgestellt werden, welcher der zuvor definierten Webseiten die aktuell angezeigte entspricht um es letztlich dem Wrapper zu ermöglichen die für diese Art von Webseite vorgesehenen Mechanismen zu aktivieren und die Navigation bzw. Extraktion fortzusetzen. Damit soll eine Verbesserung der Fähigkeiten eines Wrappers mit Unregelmäßigkeiten auf Navigationsebene umzugehen erreicht werden.

2.2.2 Extraktion

Tools und Anwendungen die sich mit der Extraktion von Daten aus dem Internet beschäftigen gibt es heute in verschiedensten Varianten, mit Umsetzung diverser Techniken und unterschiedlichen Zielen. Dieser Abschnitt gibt einen kurzen Überblick darüber und stellt einige Methoden der Datenextraktion vor. Die Arbeiten von C. Chang et. al. [4] sowie A. H. F. Laender et. al. [5] vergleichen viele verschiedene Systeme und Applikationen zur Datenextraktion aus dem Internet und geben einen guten Überblick über eingesetzte Technologien und Techniken.

Eine der größten Herausforderung in der Informationsextraktion aus dem WWW liegt darin die gesuchten Daten auch dann noch erfolgreich zu extrahieren, wenn die betreffende Webseite von ihren Betreibern verändert wurde. Unabhängig davon welche Technologien bei der Extraktion von Daten zum Einsatz kommen muss eines der Ziele des Entwicklers von Extraktionsregeln immer sein das Auslesen der gewünschten Daten möglichst unabhängig von Änderungen im Aufbau der Webseite zu machen.

Im Fall von Extraktionen mittels XML-Path Abfragen [21] wie sie auch in der im Zuge der Arbeit erstellen Applikation verwendet werden sind die Arbeiten [22] und [23] hervorzuheben, die Mechanismen vorstellen wie Extraktionsabfragen mit maximaler Robustheit gegenüber Seitenänderungen erstellt werden können. Der Kern der Aussagen dieser beiden Veröffentlichungen ist, dass die Abfragen möglichst unabhängig von in der Webseite enthaltenen Strukturen zu entwerfen seien. Dabei steht man jedoch immer vor dem Tradeoff zwischen Stabilität und Geschwindigkeit. Während absolute XPath Ausdrücke (wie zum Beispiel */html/body/div[3]/strong[@id='price']*) in kürzester Zeit abgearbeitet werden können [24] haben sie jedoch den Nachteil nicht mehr

korrekt evaluiert werden zu können, wenn sich in der HTML-Struktur auf die sie sich beziehen etwas ändert. Wird beispielsweise der Tag durch <b> ersetzt so wird obiger Ausdruck nicht mehr durchführbar und es kann kein Ergebnis geliefert werden, obwohl sich für den Betrachter der Seite kein sichtbarer Unterschied durch die Änderung im HTML-Code wird feststellen lassen.

Im Gegensatz dazu könnte man auch den Ausdruck *descendant::*[@id='price']* heranziehen. Er ist unabhängig von den strukturellen Gegebenheiten eines HTML-Dokuments und findet den Tag dessen "id" Attribut den Wert "price" enthält auch wenn dessen Tagname geändert wird, oder er nach Änderungen im HTML-Code an anderer Stelle wiederzufinden ist als vorher (beispielsweise wenn eine weitere vorgelagerte Ebene eingeführt wird).

Während man ohne großen Aufwand und spezielle Kenntnisse leicht dazu in der Lage ist robuste XPath Abfragen zu entwerfen verhält es sich mit anderen eingesetzten Technologien zur Datenextraktion anders. Viele der in [4] vorgestellten Extraktionsmechanismen sind verglichen mit XPath umständlich und komplex. Die verglichenen Systeme nutzen für ihre Extraktionsmechanismen häufig reguläre Ausdrücke [25] und formale Sprachen [26] aus auf Eigenentwicklungen basierenden Grammatiken [27] und logischen Regeln [28]. Diese zu verstehen und vor allem richtig und von Seitenänderungen unabhängig umzusetzen ist oftmals wenig intuitiv und aufwändig. Insbesondere kommt dies bei der Extraktion aus HTML-Dokumenten zum Tragen da diese häufig von den anzuzeigenden Inhalten abhängige Strukturen aufweisen die höchst variabel sein können. Speziell die Möglichkeiten von XPath 2.0 sind in diesen Fällen von großem Vorteil, denn sie bieten beliebig tief verschachtelbare if-then-else Abfragen mit denen alle möglichen Varianten eines Dokuments auf die jeweils passende Art und Weise bearbeitet werden können.

Kapitel 3

Inhalts- und Seitenanalyse

Das folgende Kapitel beschreibt welche Webseiten als Grundlage zur Entwicklung des in dieser Arbeit vorgestellten Ansatzes verwendet und welche Bestandteile dieser Webseiten analysiert wurden. Darüber hinaus wird diskutiert warum gewisse Bestandteile dieser Webseiten herangezogen bzw. nicht herangezogen wurden.

3.1 Einführung

Zur Feststellung welche Bestandteile einer Webseite in einen Fingerabdruck einfließen sollen oder nicht, bzw. als "running example" um Webseiten-Inhalte zu analysieren, den vorgestellten Ansatz zu erklären und zu evaluieren werden die Anbieter lt. Tabelle 3.1 herangezogen, wobei in der ersten Spalte der Name des die Seite betreibenden Unternehmens und in der mittleren Spalte die direkten Links zur Hotelsuche der von den Firmen zur Verfügung gestellten Webseiten zu finden sind. Die dritte Spalte enthält die im vorliegenden Dokument verwendete Kurzbezeichnung. Der Einfachheit halber sind die zu Grunde liegenden Websites auf das Angebot "Hotelzimmer buchen" beschränkt. Die Einschränkung auf diese Art von Webseiten ergibt sich aus dem Umstand dass der Hotelbuchungsmarkt im Internet ein stark umkämpfter ist und es einige Webseitenbetreiber gibt die mehrere der genannten Quellen in so genannten Metasuch-Applikationen bearbeiten um ihren Kunden eine bessere Möglichkeit zu geben Online-Preise zu vergleichen.

Ziel der Arbeit ist es es einem Rechner zu ermöglichen mittels einer Java-Applikation zwei Webseiten miteinander zu vergleichen und feststellen zu können ob die beiden Webseiten der selben Kategorie entsprechen oder nicht. Dabei sollen mehrere Bestandteile der Webseiten in einer ganz bestimmten Reihenfolge analysiert und miteinander verglichen werden um aus der Ähnlichkeit der einzelnen Komponenten schließen zu können, wie ähnlich die beiden zu vergleichenden Webseiten einander in Bezug auf ihren Zweck sind.

Der Vergleich setzt sich aus mehreren aufeinander folgenden Schritten zusammen, die

Tabelle 3.1: Beispielquellen

Anbieter	**Deeplink**	**Kurzbezeichnung**
Booking.com	http://www.booking.com	Booking
Ebookers.com	http://www.ebookers.com/shop /hotelsearch	Ebookers
Expedia Inc.	http://www.expedia.co.uk/daily /shared/products/hotels/link.aspx	Expedia
HotelClub	http://www.hotelclub.com	Hotelclub
Hotels.com	http://osterreich.hotels.com/	Hotels.com
HRS - Hotel Reservation Service	http://www.hrs.de	HRS
Travelocity.com LP	http://www.travelocity.com/Hotels	Travelocity
Venere Net S.p.a.	www.venere.de	Venere

jedoch nicht alle unbedingt vollständig durchgeführt werden müssen. Die tatsächliche Reihenfolge der einzelnen Komponentenvergleiche ist an sich nicht von Relevanz, denn am Ende wird das Ergebnis aller Vergleiche gemeinsam betrachtet. Das heißt dass eine einzelne Vergleichsebene für die Erstellung des Vergleichsergebnisses nicht von den anderen gesondert betrachtet und interpretiert wird. Die implementierte Reihenfolge der Vergleiche lautet:

1. Vergleich der Unified Resource Locators - URLs

2. Vergleich der enthaltenen Formulare

3. Vergleich der HTML-Komponenten die ein ID-Attribut aufweisen

4. Vergleich der drei höchsten Ebenen in der HTML-Baum-Teilstruktur des HTML-Bodies

Die prototypische Implementierung fügt sich in ein Wrapper-System ein, da ein Wrapper im Allgemeinen immer das selbe Set von Webseiten auf die immer gleiche Weise zu bearbeiten hat. Das heißt dass die Applikation dazu in der Lage sein muss die URL einer Webseite als Input anzunehmen und dann einen dieser Webseite entsprechenden Fingerabdruck im so genannten *Pool* von Fingerabdrücken identifizieren zu können. Die Identifizierung ist anhand obiger Komponenten einer Webseite umgesetzt und auf jeder Vergleichsebene werden je nach Grad der Ähnlichkeit Punkte vergeben. Am Ende wird jener Fingerabdruck aus dem *Pool* als dem Input-Fingerabdruck entsprechend definiert, der die höchste Punktezahl aufweist. Wie Fingerabdrücke erstellt und miteinander verglichen, und wofür wieviele Punkte vergeben werden wird in späteren Abschnitten beschrieben.

3.2 Analyse der Quell-Webseiten

Alle verwendeten Quellen haben den Zweck Konsumenten mit Informationen zu (verfügbaren) Zimmern in Hotels auf der ganzen Welt zu versorgen und es den Benutzern zu ermöglichen über diese Webseiten Hotelzimmer zu reservieren. Die Vorgehensweise ist dabei für die Kunden stets die selbe:

1. Befüllen eines Formulars mit den gewünschten Suchparametern,
2. Auswahl eines geeigneten Hotelzimmers von einer Seite die verfügbare Hotels mit Angeboten auflistet, und
3. Buchen des Angebots über eine Details zum Zimmer (und evtl. auch Hotel) enthaltenden Seite.

In dieser Arbeit wird in weiterer Folge der erste Fall als *Suchseite*, die zweite Kategorie als *Ergebnisseite* oder auch *Übersichtsseite* und der letzte Seitentyp als *Detailseite* bezeichnet werden. Darüber hinaus gibt es noch verschiedene weitere Unterkategorien von Webseiten die von Fall zu Fall vorhanden sein können wie zum Beispiel eine Seite mit Auswahloptionen zur Destination bei Booking, (Abb. 3.1).

Abbildung 3.1: Booking Stadtauswahl Ausschnitt

Um jedoch bei den in allen genützten Quellen auch vorhandenen Seitenkategorien zu bleiben setzt sich diese Arbeit nur mit Webseiten aus den drei zuvor genannten Kategorien auseinander. Auch zur tatsächlichen Buchung eines Zimmers notwendige weitere Webseiten und andere Möglichkeiten wie die Eingabe weiterer Optionen wie Mietwagen, Flughafentransfer und ähnliches sowie weitere Zusammenfassungen vor der verbindlichen Buchung und der Dialog zur Eingabe von Kundendaten und Zahlungsweise und dgl., sowie aufgrund von fehlerhaften Benutzereingaben angezeigte Seiten bleiben unberücksichtigt. Im Szenario der Datenextraktion zum Preisvergleich von Hotelzimmern sind diese Seitenkategorien weder für Navigations- noch Extraktionsschritte von

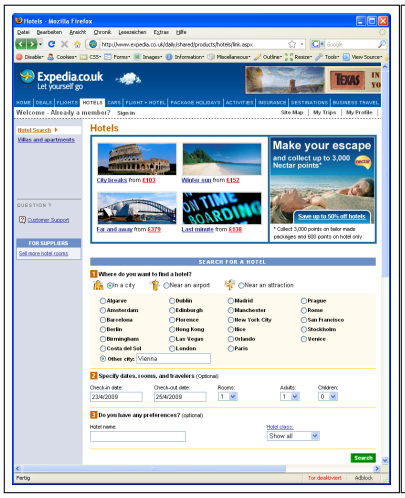


Abbildung 3.2:
Suchseite

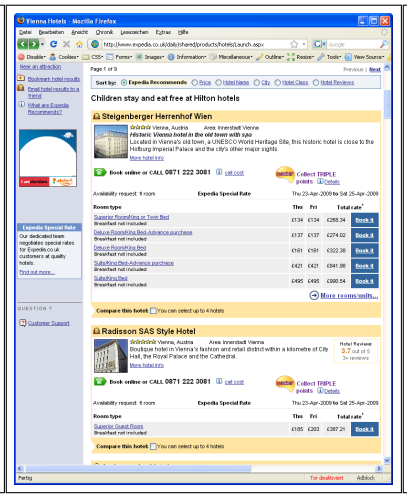


Abbildung 3.3:
Ergebnisseite

Abbildung 3.4:
Detailseite

Relevanz, denn die wichtigsten und interessantesten Daten lassen sich in den drei berücksichtigten Kategorien finden. Zu den Daten von allergrößtem Interesse zählen im Allgemeinen der Preis eines Angebots, sowie die Zimmer- und auch die Hotelkategorie die in allen Quellen entweder auf der Übersichts- oder der Detailseite zu finden sind.

Zur Illustration zeigen die Abbildungen 3.2 - 3.4 (©Expedia Inc.) die Darstellungen der erwähnten drei Schritte anhand der *Expedia* Website.

Die in Tabelle 3.1 aufgelisteten Quell-Websites dienen im ersten Schritt dazu festzustellen welche der in der folgenden Auflistung enthaltenen Inhalte dazu geeignet erscheinen zur Erstellung eines Fingerabdrucks herangezogen zu werden. Im ersten Schritt werden also Analysen angestellt mit Hilfe derer die aufgelisteten Webseiten-Elemente dahingehend untersucht werden ob sie sich zur Erstellung des Fingerabdrucks einer Webseite eignen, oder ob sie dabei nicht berücksichtigt werden sollen. Auf welche Inhalte dabei eingegangen wird zeigt die folgende Auflistung:

- Uniform Ressource Locator (URL)
- HTML-form-Strukturen und deren Inhalte
- HTML-table-Strukturen
- HTML-Strukturen mit ID-Attributen
- Struktur der drei höchsten Ebenen des HTML-Body-Teilbaums

Zu Beginn wird auf die Unified Resource Locators (URLs) der Webseiten eingegangen, daran anschließend werden die strukturellen HTML-Elemente betrachtet.

3.2.1 URLs

Mittels einer URL kann man eine Webseite in einem Webbrowser [29] zur Anzeige bringen lassen. Jede Webseite hat eine eindeutige URL, was beim Vergleichen und Identifizieren von Webseiten von großem Nutzen sein sollte. Beim hier vorgestellten Webseiten-Vergleich geht es immer darum Subseiten einer bestimmten Domäne miteinander zu vergleichen und festzustellen welche der in dieser Website enthaltenen Unterseiten gerade angezeigt wird. Alle unter einer Domäne vereinten Webseiten haben URLs die einander in gewissem Maße ähnlich sind.

Der Vergleich der URLs zweier miteinander zu vergleichender Webseiten kann bereits viel darüber aussagen, ob die beiden Seiten in Bezug auf ihre Kategorie einander entsprechen oder nicht. Einige URLs enthalten eine große Anzahl an Variablen und Parametern, während andere vieles vor dem Benutzer versteckt halten (vgl. Ergebnisseiten von Travelocity oder HRS). Diese Dynamik in den Webseiten-Adressen erschwert einen Vergleich natürlich, aber grundsätzlich ist die URL einer bestimmten Webseite strukturell immer gleich aufgebaut.

Für die Bestimmung der Gleichheit zweier URLs bietet es sich an einen Longest Common Subsequence (LCS) [30] Algorithmus zu verwenden. Die LCS zweier Strings sind alle jene Characters (Buchstaben) die in beiden zu vergleichenden Zeichenfolgen vorhanden sind und zumindest einen Vorgänger aufweisen können der ebenfalls in beiden Strings enthalten ist. Dieser Vorgänger muss dabei nicht der direkt vorangestellte Buchstabe sein, sondern kann auch weiter vorne im String liegen, also einen oder mehrere Characters als Nachfolger haben, die nicht in der LCS enthalten sind. Um dies zu ermöglichen wird noch vor dem jeweils ersten Zeichen der beiden Zeichenfolgen ein null Zeichen eingefügt, wodurch die beiden Strings ab dem ersten Zeichen vergleichbar sind obwohl diese ja eigentlich keinen Vorgänger haben.

Da bei der Bestimmung der LCS die beiden zu vergleichenden URLs immer vollständig verglichen werden ist dies im vorliegenden Fall nicht die effizienteste Lösung. Die Übersichtsseite von Expedia beispielsweise hat eine URL mit einer Länge von über 850 Zeichen, während die URL der Detailseite über 300 und die Formularseite wiederum nur knapp mehr als 60 Characters enthält. LCS Algorithmen würden nun beide zu vergleichenden Strings in ihrer vollen Länge miteinander in Beziehung setzen obwohl nach nur 25 Zeichen klar ist, dass keine weiteren Gemeinsamkeiten mehr vorliegen können da die an die Domäne (http://www.expedia.co.uk) angehängten Parameter in keiner Weise überein stimmen. Dass die einzelnen Webseitenkategorien im Fall von Expedia tatsächlich bereits nach den ersten 25 Zeichen eindeutig unterscheidbar sind zeigt die folgende Aufstellung:

Formularseite:	http://www.expedia.co.uk/Hotels?action=ho[...]Input=0
Übersichtsseite:	http://www.expedia.co.uk/daily/shared/p[...]hotels/link.aspx
Detailseite:	http://www.expedia.co.uk/pub/agent.dll/qscr[...]6!4$FF
Gemeinsamer Inhalt:	http://www.expedia.co.uk/

Man kann eine der gefragten Seitenkategorien also sehr einfach, schnell und zuverlässig identifizieren indem man sich den Beginn des auf die gemeinsame Domäne folgenden Parameterteils der URL ansieht. Deshalb wird im vorliegenden Fall die Ähnlichkeit der URLs zweier zu vergleichender Webseiten bestimmt indem von links nach rechts Buchstabe für Buchstabe miteinander verglichen werden um so den längsten gemeinsamen durchgehenden Substring ab dem ersten Character zu bestimmen. Es wird die zu vergleichende URL mit den URLs aller Fingerabdrücke im Wrapper verglichen und die zu bestimmende Seite wird der Seite als am ähnlichsten zugewiesen, deren Übereinstimmung die meisten Zeichen umfasst. Diese Methode ist für die Webseiten Booking, Ebookers, Expedia, Hotelclub und auch Venere ausreichend um die drei Subseiten voneinander zu unterscheiden. Für die Seite Hotels.com, deren Links bereits im Domänen-Teil der URL direkt nach dem 'www' variabel sind und die zu verwendende Sprache, bzw. das aus der IP-Adresse abgeleitete Herkunftsland des Benutzers aufweisen, wird ein URL-Vergleich als solches nicht von großem Nutzen sein. Da allein der Vergleich ihrer URLs nicht ausreichend ist um die Gleichheit der Kategorien zweier Seiten auszudrücken müssen andere Bestandteile dazu dienen wie der Vergleich der in der Seite enthaltenen HTML-Struktur um eine eindeutige Zuweisung zu ermöglichen.

3.2.2 HTML-Struktur

Die Hypertext Markup Language [31] ist eine verschiedenste Komponenten beinhaltende Auszeichnungssprache die von Webbrowsern verarbeitet werden kann. Da die in HTML verwendeten Komponenten beschränkt sind müssen für Webseiten ähnlichen Zwecks auch ähnliche HTML-Komponenten verwendet werden, denn für verschiedene Vorhaben werden jeweils dafür vorgesehene Konstrukte zur Verfügung gestellt. Als erstes HTML-Konstrukt werden in den jeweiligen Seiten enthaltene Formulare auf ihre Tauglichkeit hinsichtlich der Erstellung eines Fingerabdrucks analysiert. Weiters werden Tabellen, beliebige mit einem ID-Attribut ausgezeichnete HTML-Tags sowie ein Teil des HTML-Baums betrachtet.

HTML-Formulare

Damit Nutzer einer Website in der Datenbank des Seitenbetreibers interessante Hotels finden können muss ihnen eine Möglichkeit gegeben werden die Inhalte der Datenbanken gemäß ihren Bedürfnissen abzufragen. Im Allgemeinen werden solche Abfragen mittels Formular-Strukturen realisiert. Allein zu wissen dass eine Webseite ein Formular enthält reicht jedoch nicht aus um eine Seite eindeutig als Suchseite identifizieren zu können. Auf Übersichts- und Detailseiten sind in allen untersuchten Fällen nämlich ebenfalls Formulare enthalten.

Neben der Anzahl der eingebundenen Formulare sind auch deren Annotationen mittels Attributwerten sowie die Inhalte der enthaltenen Formularelemente für die Erstellung eines Fingerabdrucks von Bedeutung. Die Attribute können darüber Auskunft geben

welche Aufgabe ein Formular hat, und die Elemente darüber welche Inhalte. Insbesondere gilt dies für jene Quellen bei denen Suchergebnisse (Booking) oder Details zu Angeboten (Hotels.com und Venere) in Formularen festgehalten sind. Jedes Formular kann eine Vielzahl von Attributen besitzen [1] von denen nur *method* und *action* verpflichtet gesetzt sein müssen. Für die Analyse von Interesse sind die Attribute *ID* sowie *name*, die das Formular-Objekt beschreibende Informationen enthalten, und *action*, das anführt wohin die in den Formular-*Elementen* enthaltenen Informationen durch Absenden des Formulars weitergereicht werden. Mit dem Wissen um Attribute und Inhalte von Formularen kann man bereits Annahmen zur momentan angezeigten Seite treffen. Ein Formular das als Action Attribut einen Wert wie "book" oder "booking" enthält und damit dazu dient Informationen zu einem bestimmten Angebot eines konkreten Hotels zusammenzufassen und an die nächste Webseite zur Buchung weiterzuleiten, wird mit allergrößter Wahrscheinlichkeit in einer Seite vorhanden sein die erst nach der Eingabe aller Parameter angezeigt werden kann. Außer den genannten Attributen von Formular-Objekten fließen auch die in einem Formular enthaltenen Elemente in einen Fingerabdruck mit ein. Berücksichtigt werden die Anzahl der enthaltenen Elemente sowie deren Inhalte in den Attributen *name* und *ID*.

Es existieren Formulare die in ihren Attributen ID, Name und Action nicht völlig gleich, einander aber sehr ähnlich sind und auch die selbe Aufgabe haben. Beispielsweise listet Booking seine Ergebnisse auf der Ergebnisseite mit Hilfe von Formularen auf, und jedes angezeigte Suchergebnis ist in einem Formular-Konstrukt eingeschlossen. Alle diese Formulare haben die Aufgabe den Kunden auf die Detailseite zu diesem Hotel weiterzuleiten und alle haben sehr ähnliche bis gleiche Inhalte in ihren verglichenen Attributen. Im Fall von Booking ist nur das Action Attribut mit folgender Struktur gesetzt, während die beiden Attribute Name und ID völlig leer sind:

/hotel/*{Länder-Code}*/*{Hotel-Name}*.html
?sid=*{Session-ID}*;checkin=*{Checkin}*;checkout=*{Checkout}*

Der Vergleich basierend auf diesem Attribut ist nicht zielführend da der einzige allen Formularen gemeinsame Teil des ID-Attributs das zu Beginn stehende /hotel/ ist. Es bestünde die Möglichkeit die Inhalte der Action-Attribute um ihre offensichtlich dynamischen Inhalte - also Substrings zwischen "=" und dem darauf folgenden ";" - zu reduzieren. Damit würde man zwar die Strings einander vergleichbarer machen, jedoch bliebe ein Rest von Variabilität erhalten, da noch im link-Pfad vor den Parametern "Länder-Code" und "Hotel-Name" des aufzurufenden Hotels enthalten sind.

Um auch Formulare die nicht die exakt selben Attribute tragen als einander entsprechend identifizieren zu können werden sie aufgrund ihrer enthaltenen Elemente verglichen. Wenn es ein Formular in einer Webseite gibt dessen Inhalte an eine im Action-Attribut definierte Webseite weitergeleitet werden sollen so gibt es immer auch Formular-Elemente, die die weiterzuleitenden Informationen beinhalten.

Zwei Formulare werden als einander entsprechend definiert, wenn sie:

[1] siehe http://www.w3.org/TR/1999/REC-html401-19991224/interact/forms.html#h-17.3 für die vollständige Liste

1. Die selbe Anzahl an Elementen enthalten.
2. Alle enthaltenen Elemente in Name und ID übereinstimmen.
3. Sie in zwei der drei Formular-Attribute Name, ID und Action übereinstimmen.

Warum nur zwei der drei untersuchten Attribute gleich sein müssen begründet sich darin, dass, wie zuvor bereits beschrieben, die Action Attribute von Formularen häufig variable Inhalte aufweisen. Um die Gleichheit zweier Formulare zu bestimmen würde die Einbeziehung aller drei Attribute in den Vergleich sehr häufig darauf hinauslaufen, dass zwei Formulare die die selben Elemente besitzen und faktisch die selbe Aufgabe haben nicht als "gleich" identifiziert werden würden. Aus dem selben Grund wird die Untersuchung der Formular-Attribute erst nach der Analyse auf Elementebene durchgeführt. Würde man zu Beginn des Vergleichs alle drei berücksichtigten Attribute zweier Formulare in Beziehung setzen wäre im Fall der Booking-Übersichtsseite keine Ähnlichkeit zwischen den einzelnen in Formular-Konstrukten eingeschlossenen Angeboten festzustellen, was nicht korrekt wäre. Diese Reihenfolge von Vergleichen bezieht sich jedoch nur auf den Formularvergleich, es ist sozusagen eine "interne" Reihenfolge. Ob die URLs oder HTML-Tags mit ID-Attribut vor oder nach den Formularen verglichen werden steht damit nicht in Zusammenhang, diese Abfolge der einzelnen Komponenten ist für das Vergleichsergebnis bedeutungslos.

Ein wesentlicher Unterschied zwischen den einzelnen Anbietern und auch den Anbieterinternen Seitentypen besteht unter anderem in der Anzahl der enthaltenen Formulare. Tabelle 3.2 listet die Anzahl der in den jeweils betrachteten Subseiten enthaltenen Formularstrukturen auf. Dies soll ihre Verwendung untermauern, denn wie man sieht sind sie verglichen mit eingebetteten Tabellen in sehr kleinen und auch innerhalb der selben Seitenkategorie gleich bleibender Anzahl vorhanden. Selbst wenn die Inhalte verschiedener Varianten der selben Seitenkategorie sehr unterschiedlich sind - beispielsweise wenn ein Hotel nur zwei Zimmer anbietet, ein anderes jedoch mehr - gibt es immer zumindest ein Formular das unabhängig von der Anzahl der angezeigten Angebote vorhanden ist und damit die Wiedererkennung ermöglicht. Darüber hinaus findet man in jeder der verwendeten Beispielseiten zumindest ein Formular.

Tabelle 3.2: Anzahl enthaltener Formulare

Anbieter	**Suchseite**	**Übersichtsseite**	**Detailseite**
Booking	4	# Hotels + 4	7
Ebookers	1	1/2	1
Expedia	1	5	2
Hotelclub	4	5/4	7
Hotels.com	3	4	# Angebote + 1
HRS	3	9	2
Travelocity	4	5	2
Venere	2	2	# Angebote + 1

Booking enthält auf jeder verfügbaren Übersichtsseite 15 buchbare Hotels, außer es

wurden weniger als 15 für die eingegebenen Such-Parameter gefunden. Werden mehr als 15 Ergebnisse geliefert und ist diese Anzahl nicht ohne Rest durch 15 teilbar befinden sich auf der letzten Übersichtsseite zwischen einem und 14 Hotels. Dadurch kann es zu unterschiedlichen Anzahlen von Formularen kommen, denn auf jeder Übersichtsseite befinden sich vier Formulare plus der Anzahl der angezeigten Ergebnisse. Also 19 Formulare für eine volle Ergebnisseite und beispielsweise acht wenn nur vier verfügbare Hotels angezeigt werden. Die vier Formulare die unabhängig von der Ergebnisanzahl vorhanden sind ermöglichen es die Suchparameter abzuändern und die Ergebnisse neu generieren zu lassen, die für die Textinhalte verwendete Sprache sowie die angezeigte Währung zu ändern und die Sortierung der Ergebnisse anzupassen. Darüber hinaus existiert noch ein aus nur einem Feld bestehendes, verstecktes Formular dessen Zweck nicht ersichtlich ist.

Hotelclub liefert für eine Anfrage eine Liste mit verfügbaren und auch nicht verfügbaren Hotels, dazu kommen auf der ersten Seite mit Suchergebnissen noch einige mit "Most Popular" gekennzeichnete Hotels wodurch in der ersten Übersichtsseite fünf und in allen weiteren Übersichtsseiten vier Formulare enthalten sind. Detailseiten von Hotels.com enthalten für jedes auf ein bestimmtes Hotel bezogenes Angebot ein eigenes Formular, demnach finden sich auf einer Detailseite mit vier verschiedenen Zimmerangeboten auch vier Formulare im HTML-Code. Darüber hinaus enthält die Webseite noch ein weiteres Formular in dem Hotel- und Reisedaten zusammengefasst sind und das es dem Nutzer ermöglicht seine Suchparameter zu ändern und die Ergebnisse neu generieren zu lassen. Gleich verhält es sich auf den Detailseiten von Venere, wo zur Anzahl der Angebote ebenfalls noch ein weiteres Formular hinzukommt mit dem sich die Suchparameter ändern und die Ergebnisse für dieses Hotel dementsprechend aktualisieren lassen.

Wie sollen Formulare nun in den Fingerabdruck einer Webseite einfließen? Zunächst werden alle in der Webseite gefundenen Formulare ausgelesen und ihre Attribute ID, Name und Action sowie die Namen und IDs aller vom Formular zu übertragenden Elemente festgehalten. Wie Tabelle 3.2 zeigt gibt es bei den Seiten Booking, Hotel.com und Venere möglicherweise eine große Zahl von enthaltenen Formularen die einander äußerst ähnlich sind da sie im Grunde gleiche Objekte auflisten und sie alle auch die selbe Aufgabe haben. Um zu verhindern dass die Anzahl an solchen gleichen Formularen in einer Seite den Vergleich beeinflusst werden alle Formulare einer Seite bevor sie in den Fingerabdruck einfließen mit allen anderen in der Seite enthaltenen Formularen verglichen. Dadurch werden einander sehr ähnliche Formulare identifiziert und ausgeschieden; eines dieser Formulare wird jedoch als Repräsentant behalten und mit einem Hinweis versehen dass es ursprünglich mehr als nur einmal enthalten war. Behalten wird immer das als erstes gefundene Formular, also jenes das in der Webseite visuell am weitesten oben zu finden ist. Die Position des behaltenen Formulars in der Reihe der vorkommenden und als gleich eingestuften Formulare ist nicht von Bedeutung. Jedes dieser gleichen Formulare wäre ein gleich guter Repräsentant wie das erste.

Um diesen internen Vergleich zu ermöglichen müssen zuvor jedoch noch die Elemente entsprechend behandelt werden. Hotelclub hat auf seiner Übersichtsseite ein Formular in dem alle aufgelisteten Angebote enthalten sind. Dadurch kann eine sehr große Anzahl

gleicher Elemente zustande kommen, die einander in den Attributen Name und ID entsprechen. Hier werden ebenso alle bis auf ein repräsentatives und auch als solches gekennzeichnetes Element aus dem Formular ausgeschieden. Durch diese Maßnahme werden alle mehr als einmal vorkommenden Elemente aus der Datenstruktur entfernt und es ermöglicht Formulare die ursprünglich die selben Elemente in unterschiedlicher Anzahl enthalten hatten korrekt miteinander zu vergleichen, unabhängig davon wie viele solcher mehrfach vorkommenden Elemente tatsächlich enthalten waren.

HTML-Tabellen

Tabellen eignen sich gut zur strukturierten Darstellung von Inhalten. Im Gegensatz zu Formularen haben Tabellen jedoch keine Attribute die verpflichtend gesetzt sein müssen [2]. Dadurch kämen in einem Fingerabdruck nur Tabellen in Frage die gesetzte Attribute aufweisen um sie mit deren Hilfe eindeutig identifizieren zu können. Im Gegensatz zu den Formularen sind jedoch nicht in jeder Seitenkategorie aller Quellen Tabellen enthalten. Sie eignen sich dadurch also nicht als primäre, also sehr aussagekräftige und eindeutige Vergleichsobjekte, sondern sollten erst dann in den Vergleich mit einbezogen werden, wenn sich aus anderen Vergleichsmethoden kein eindeutiges Ergebnis ergibt und in der jeweiligen Seite Tabellen mit zumindest gesetztem ID-Attribut vorhanden sind.

Häufig kommt es vor, dass Tabellen ineinander verschachtelt sind, das heißt, dass eine Tabelle eine oder mehrere weitere Tabellen enthält die wiederum ebenfalls Tabellen enthalten können. Diese Komplexität und die Tatsache dass Seiten sehr viele Tabellen enthalten können lässt die Tabellen-Konstrukte zur Erstellung von Fingerabdrücken und deren Vergleich ausscheiden. Je mehr Objekte bei der Erstellung oder dem Vergleich von Fingerabdrücken berücksichtigt werden müssen, desto länger dauert es auch diese zu verarbeiten. Wie viele Tabellen in einer einzigen Webseite enthalten sein können zeigt Tabelle 3.3. Wie man sehen kann sind die enthaltenen Tabellen-Konststrukte nicht nur von höchst unterschiedlicher Anzahl von Anbieter zu Anbieter, sondern auch Seiten-intern gibt es große Unterschiede. So enthält die Expedia Übersichtsseite abhängig von der Anzahl der dargestellten Suchergebnisse zwischen zwei und über 200 Tabellen. Darüber hinaus kommen auch Seiten vor, die gar keine Tabellen enthalten (Formularseite von Hotelclub, HRS und Venere). Aus diesen Gründen wurde auf die Miteinbeziehung von HTML-Table Konstrukten in die Erstellung und den Vergleich von Fingerabdrücken verzichtet. Wie in Kapitel 6 beschrieben lässt sich der Vergleich auch ohne die Berücksichtigung von Tabellen erfolgreich durchführen und bestätigt somit die Entscheidung Tabellen nicht in Fingerabdrücke mit einfließen zu lassen.

[2] http://www.w3.org/TR/html401/struct/tables.html#h-11.2

Tabelle 3.3: Anzahl enthaltener Tabellen

Anbieter	Suchseite	Übersichtsseite	Detailseite
Booking	9	# Hotels + 6	13
Ebookers	4	2	4
Expedia	4	2 - 200+	42/43
Hotelclub	0	9 - 17	# Angebote + 11
Hotels.com	3	# Angebote * 4 + 5	4
HRS	0	# Angebote	1
Travelocity	1	1 - 70+	5/6
Venere	0	15 - 20+	3

HTML-Tags mit ID-Attribut

Als weitere Vergleichsebene kommen im HTML-Code vorhandene ID-Attribute beliebiger HTML-Konstrukte in Frage. Tabelle 3.4 enthält eine Übersicht über die Anzahl der in den einzelnen Webseiten enthaltenen HTML-Tags mit ID-Attributen. Die Zahlen sind jedoch nur ungefähre Werte und nicht immer absolut korrekt, da sich ID-Tags abhängig von der Anzahl angebotener Zimmer u. dgl. in der Seite befinden können, die bei der Analyse nicht in jedem Fall vorhanden waren. Diese Abweichungen bewegen sich jedoch in keiner bedeutenden Bandbreite und sind somit zu vernachlässigen.

Mittels geeigneter XPath-Ausdrücke und Java-Methoden (siehe Kapitel 4.1.3), lassen sich alle in einer Seite enthaltenen HTML-Konstrukte finden die ein Attribut namens "ID" besitzen. Zur weiteren Bestimmung der Ähnlichkeit zweier Webseiten werden die enthaltenen HTML-Tags und deren ID-Attribute miteinander verglichen.

ID-Attribute sind in jeder der Beispielseiten vorhanden, jedoch oft in sehr unterschiedlicher Häufigkeit. Meist sind sie auch an die dargestellten Hotels, bzw. deren angebotene Zimmer gekoppelt und enthalten eine das Hotel bzw. das Zimmer, das Angebot, die inkludierte Verpflegung usw. eindeutig identifizierende Bezeichnung oder Nummer. Solche Inhalte sind für den Vergleich zwar hinderlich, verhindern ihn jedoch nicht vollständig. Neben solchen von den dargestellten Inhalten abhängigen IDs sind nämlich in jeder Seite auch HTML-Tags mit ID-Attributen vorhanden die unabhängig von den tatsächlich angezeigten Inhalten und immer in der jeweiligen Seitenkategorie zu finden sind. Diese spezifischen Elemente der einzelnen Kategorien reichen aus um den Vergleich neben der Verwendung der URLs und Formulare weiter zu verbessern und genauer zu gestalten.

Wie bereits erwähnt ist die Anzahl der in einer bestimmten Seitenkategorie enthaltenen IDs durchaus variabel. Booking beispielsweise hat auf der Übersichtsseite zu jedem Angebot auch noch eine Information zur Verfügbarkeit des Zimmers bzw. der Zimmer-Kategorie angeführt. Je nach Verfügbarkeit des Zimmers wird es dann auch mit zwei oder drei HTML-Tags mit ID-Attribut, die sich mittels einer Kennnummer auf das entsprechende Hotel und einer weiteren daran angehängten zweistelligen Num-

mernkombination auf Zimmerkategorie und Verfügbarkeit beziehen im HTML-Code repräsentiert. Die Anzahl der unabhängig von den angezeigten Hotels und Angeboten in der Seite vorhandenen HTML-Tags mit ID-Attribut beläuft sich auf ca. 80.

Die Anzahl der in einer Detailseite von Booking enthaltenen ID-Attribute ist einerseits abhängig von der Anzahl der gelisteten Angebote, sowie andererseits davon wie viele Kommentare früherer Gäste des Hotels sich auf der Seite befinden. Unabhängig von den dargestellten Inhalten gibt es 145 HTML-Tags die ein ID-Attribut tragen.

Ebookers verwendet auf der Formularseite 16 HTML-Tags mit ID-Attribut wenn keine Suchvorschläge aufgrund vergangener Abfragen angezeigt werden; in diesem Fall wären es 19. Auf der Übersichtsseite ist die Anzahl der enthaltenen IDs nicht von der Anzahl der angezeigten Hotels oder Angebote abhängig, sondern davon wieviele Ergebnisfilter möglich sind. Für jede Möglichkeit die Ergebnisse nach bestimmten Eigenschaften zu filtern (zum Beispiel nur Hotels mit inkludiertem Frühstück oder gratis Abholservice vom Flughafen usw.) existiert ein HTML-Konstrukt mit einem ID-Attribut. Darüber hinaus gibt es noch drei IDs mit einem Hashkey über den die Benutzer der Seite identifiziert werden können. Abgesehen davon existieren etwa 25 für den Vergleich relevante IDs in dieser Seitenkategorie von Ebookers.

Die weiteren Quellseiten verhalten sich ganz ähnlich und haben in vielen Fällen recht variable Mengen an enthaltenen HTML-Tags mit ID-Attributen. Tabelle 3.4 dient also dazu einen ungefähren Eindruck über die relevanten Inhalte und die Größenordnungen zu geben.

Tabelle 3.4: Durchschnittliche Anzahl enthaltener Tags mit ID-Attributen

Anbieter	**Suchseite**	**Übersichtsseite**	**Detailseite**
Booking	164	350	200
Ebookers	16	15	12
Expedia	400	100	160
Hotelclub	92	53	45
Hotels.com	37	70	65
HRS	63	210	85
Travelocity	116	90	75
Venere	64	360	175

HTML-Teilbaum

Als letzte Vergleichsebene wird ein stets vorkommender Teil des einer Webseite zu Grunde liegenden HTML-Baumes [32] herangezogen. Dabei werden die zwei dem Tag <body> folgenden Ebenen berücksichtigt wodurch sich eine aus drei Ebenen bestehende Teilbaum-Struktur ergibt. Dieser Teilbaum der verwendeten Quellen wurde gewählt weil sich in den Analysen der Quell-Webseiten gezeigt hat, dass er sich von Seitenkategorie zu Seitenkategorie stark unterscheidet bzw. innerhalb des selben Typs von

Webseite recht stabil ist. Es wurde festgestellt, dass Webseiten der selben Kategorie und der selben Domäne in den berücksichtigten Strukturen zwar nicht in jedem Fall absolut gleich sind, die Unterschiede zu anderen Kategorien jedoch so groß sind, dass eine korrekte Unterscheidung mit allergrößter Wahrscheinlichkeit möglich ist. Das bedeutet, dass zwei Webseiten aus unterschiedlichen Kategorien beim Vergleich der HTML-Teilbäume mit an Sicherheit grenzender Wahrscheinlichkeit nicht als gleich eingestuft werden. Zwar werden zwei einander entsprechende Webseiten beim Vergleich nicht in jedem Fall über diesen Teilbaum einander zugewiesen werden können, eine falsche Zuweisung ist jedoch auszuschließen.

Die Darstellungen der einzelnen Subseiten der berücksichtigten Quellen weisen stets markante und für einen Menschen sofort sichtbare Unterschiede auf die es ermöglichen auf den ersten Blick eine bereits bekannte Art von Webseite wiederzuerkennen. Diese Unterschiede schlagen sich bereits auf höchster Ebene des durch einen Browser darstellbaren Bereichs des HTML-Codes einer Webseite nieder. Nachdem alles was in einem Browser für den Benutzer sichtbar angezeigt werden soll im Tag <body> seine Wurzel haben muss, liegt es nahe diesen Bereich des HTML-Codes in die Fingerabdruck-Repräsentation einer Webseite mit aufzunehmen. Und da der Inhalt des <body> Knotens sehr umfangreich sein kann beschränken wir uns bei seiner Berücksichtigung auf die ersten beiden dem <body> folgenden Ebenen. Wie erwähnt sind diese Strukturen zwar nicht immer in jeder Seitenkategorie absolut unveränderlich, jedoch kann man in den Testergebnissen des Kapitels 6 sehen, dass ein korrekter Vergleich in vielen Fällen auch durch diesen Teil der Fingerabdrücke unterstützt wird.

In den Fingerabdruck fließt damit die dem <body> Element folgende Knotenebene vollständig ein. Von jedem direkten Nachfolger des body-Knotens werden der Name sowie die Position innerhalb seiner Geschwisterknoten festgehalten. Darüber hinaus werden auch die diesen Knoten folgenden Kindknoten (also vom <body>-Tag aus gesehen der zweiten Schachtelungsebene) auf gleiche Weise berücksichtigt. Damit ist die vollständige Struktur des HTML-Baumes des <body> Elements mit seinen zwei folgenden Ebenen abgebildet.

Kapitel 4

Erstellung und Vergleich von Fingerabdrücken

Die im vorigen Kapitel diskutierten Bestandteile von Webseiten sollen dazu herangezogen werden eine Repräsentation einer Webseite zu bilden. Die Webseiten-Bestandteile URL, Formulare, HTML-Tags mit ID-Attributen und der erwähnte HTML-Teilbaum werden in einem Fingerprint-Objekt gespeichert und dienen dazu den Vergleich zweier Webseiten zu ermöglichen indem die Fingerabdrücke der beiden Seiten miteinander in Beziehung gesetzt werden. Dieses Kapitel beschreibt nun die beiden wichtigsten Funktionalitäten der erstellten Applikation, nämlich die Abläufe um einen Fingerabdruck zu erstellen, sowie die Vorgehensweise beim Vergleich zweier Fingerabdrücke miteinander.

4.1 Erstellung von Fingerabdrücken

Von jeder Webseite deren Fingerabdruck erstellt werden soll sind zu Beginn lediglich die URL und der zu vergebende Name bekannt. Mit Hilfe der URL und einer verfügbaren Internetverbindung werden die Inhalte der Webseite an die Applikation übergeben und so alle weiteren Schritte ermöglicht. Wie dies für die einzelnen Bestandteile konkret umgesetzt wurde wird in den folgenden Abschnitten beschrieben.

4.1.1 URLs

Die URL der Webseite deren Fingerabdruck erstellt werden soll muss zur Erstellung des Fingerabdrucks als Input an die Applikation übergeben werden. Neben dem ebenfalls als Startparameter übergebenen Namen des Fingerabdrucks wird die URL auf Programmiersprachen-Ebene als ein Attribut eines Fingerabdruck-Objekts gespeichert. Alle weiteren Fingerabdruck-Bestandteile können erst dann be- bzw. verarbeitet werden wenn die URL erfolgreich aufgerufen und die von ihr repräsentierte Webseite erfolgreich an die Applikation übertragen wurde.

4.1.2 Formulare

Nachdem die URL aufgerufen und der HTML-Code erfolgreich an die Applikation übertragen ist wird er mit Hilfe einer Mozilla Programmbibliothek (Details dazu siehe Abschnitt 5.2) in ein DOM Dokument umgewandelt. Damit entspricht das ursprüngliche HTML-Dokument nun den Regeln eines XML-Dokuments [33] und es ist möglich die Inhalte mittels XPath-Ausdrücken [21] weiter nach seinen Vorstellungen zu bearbeiten und zu filtern. Dadurch wird es ermöglicht den XPath Ausdruck "//form" abzusetzen, dessen Ergebnis ein Objekt ist, das alle <form> Konstrukte und deren (weitere) Inhalte wie Elemente und Attribute enthält. Damit sind alle im HTML-Code der Seite enthaltenen Formulare zur weiteren Behandlung im System verfügbar.

Da in einem Fingerabdruck auch die von jedem Formular zu übergebenden Elemente benötigt werden, muss der XPath Ausdruck ".//input" abgesetzt werden. Diese Abfrage liefert alle HTML-Elemente die <input> heißen und zwischen den HTML-Tags <form> und </form> jedes zuvor gefundenen Formulars liegen. Die auf diese Weise gefundenen Elemente werden dem jeweiligen Formular zugewiesen; bevor sie jedoch tatsächlich in ein Formular-Objekt einfließen werden sie auf Duplikate überprüft. Wie beispielsweise in der Übersichtsseite von Hotelclub kann es vorkommen, dass ein Formular eine Vielzahl von Elementen enthält die alle den selben Namen und die selbe ID tragen. Im Fall der Übersichtsseite von Hotelclub exisitert ein Formular das alle für die Suchparameter verfügbaren Hotels enthält, was bei bis zu 40 angezeigten Ergebnissen 40 gleichen Einträgen entspricht. Jedes Element repräsentiert ein Hotel dessen Detailinformationen über einen Button aufgerufen werden können. Zwei in einem Formular enthaltene Elemente die in Namen und ID übereinstimmen werden als gleich angesehen und es werden alle Duplikate entfernt. Ein Element bleibt jedoch im Formular erhalten und bekommt den Zusatz, dass es im Original mehrfach vorgekommen ist.

Sind die Elemente analysiert und Duplikate entfernt werden sie in einem *Vector*-Objekt gespeichert und bilden gemeinsam mit den Attributen Name, ID und Action des beinhaltenden Formulars ein gemeinsames java-Objekt vom Typ *Form*.

Daran anschließend werden all diese zuvor gemeinsam in einer Webseite enthaltenen Formulare ebenfalls auf Duplikate überprüft. Dies ist notwendig da zum Beispiel in der Booking-Übersichtsseite (siehe Tabelle 3.2) die Anzahl der enthaltenen Formulare von der Anzahl der gefundenen Hotels abhängt. Auch hier werden wieder Duplikate ausgeschieden. Es werden also in einer Webseite enthaltenen Formulare miteinander verglichen, und dies geschieht in folgender Reihenfolge:

1. Anzahl enthaltener Elemente vergleichen.
2. Namen und IDs der Elemente vergleichen.
3. Formular-Attribute vergleichen

Nur wenn zwei Formulare die selbe Anzahl an enthaltenen Elementen aufweisen werden sie weiter miteinander verglichen um festzustellen ob die enthaltenen Elemente nicht nur in ihrer Anzahl übereinstimmen, sondern auch ob deren Namen und IDs einan-

der entsprechen. Ist dies ebenfalls gegeben erfolgt noch eine letzte Überprüfung der Formular-Attribute. In Betracht gezogen werden Name, ID und Action jedes Formulars. Stimmen zwei Formulare neben ihren enthaltenen Elementen auch noch in zwei der drei Attribute überein, werden sie als gleich angesehen. Attribute können Strings enthalten oder auch *null* sein.

Finden sich in einer Webseite Duplikate eines gewissen Formulars werden diese dem Fingerabdruck der Webseite nicht hinzugefügt. Es wird nur ein Formular mit seinen Attributen und den enthaltenen Elementen sowie dem Zusatz dass es ursprünglich mehrfach in der Seite vorhanden war in den Fingerabdruck aufgenommen. Die technische Umsetzung dieser Vorgehensweise zeigt der Codeausschnitt 4.1.

Diese Methode organisiert den Vergleich aller in einer Webseite enthaltenen Formulare miteinander, die in einem an sie übergebenen *FormContainer* Objekt enthalten sind. Sie ruft die Methoden auf die den detaillierten Vergleich umsetzen und gibt ein *FormContainer* Objekt zurück das keine mehrfach und als gleich zu betrachtenden *Form*-Objekte mehr enthält.

Listing 4.1: compareFormsWithinPage

```
1  public FormContainer compareFormsWithinPage(FormContainer
       formContainer) {
2   Vector<Form> compareForms = formContainer.getForms();
3   Vector<Form> formsToCompareWith = formContainer.getForms();
4   FormContainer comparedForms = new FormContainer();
5    for(int i = 0; i<compareForms.size(); i++) {
6      Form compareForm = compareForms.get(i);
7      if(!compareForm.isMultiple()) {
8        for(int j = i+1; j<formsToCompareWith.size(); j++) {
9          Form formToCompareWith = formsToCompareWith.get(j);
10         if(!formToCompareWith.isMultiple()) {
11           if(compareForm.getNumberOfElements() ==
                formToCompareWith.getNumberOfElements()) {
12             compareElementsOfTwoForms(compareForm,
                  formToCompareWith, "internal");
13       } } }
14     comparedForms.addForm(compareForm);
15   } }
16  return comparedForms;
17 }
```

Der erste Teil des Formularvergleichs wird im obigen Listing in Zeile 12 angestoßen indem die Methode *compareElementsOfTwoForms(Fingerprint fp1, Fingerprint fp2, String flag)* aufgerufen wird, deren Aufgabe darin besteht die Elemente zweier Formulare zu vergleichen und festzustellen ob in den beiden Formularen gleiche Elemente enthalten sind. Diese Methode wird nicht nur beim seiteninternen Vergleich verwendet, sondern auch beim Vergleich zweier Fingerabdrücke miteinander.

Nachdem Element-Attribute entweder einen *String* Wert enthalten oder auch *null* sein können muss beim Vergleich der Werte jeweils auf diese Tatsache Rücksicht genommen und der Vergleich daran angepasst umgesetzt werden. Jedes Element kann vier verschiedene Kombinationen an String bzw. Null-Werten in seinen Attributen haben, und für jede dieser Kombinationen existieren im folgenden Codestück zwei if-Abfragen, die diese bei den zu vergleichenden Elementen überprüfen um die Attribute korrekt vergleichen zu können.

Wie auch im vorherigen Codeausschnitt ersichtlich werden an die Methode in Listing 4.2 die zwei zu vergleichenden *Form*-Objekte sowie ein String übergeben. Der String "caller" enthält den Wert "internal" was zum Ausdruck bringen soll, dass Formulare aus der selben Webseite miteinander verglichen werden sollen und wird in Zeile 34 an die Methode *compareAttributesOfTwoForms(compareForm, formToCompareWith, requester)* weitergereicht in der die Attribute der Form-Objekte verglichen werden. Dieser Aufruf findet jedoch nur dann statt, wenn der Wert der Hilfsvariable *sameElementCounter* der Anzahl der enthaltenen Elemente entspricht was bedeutet, dass alle Elemente gleich sind.

Listing 4.2: compareElementsOfTowForms

```
1  private int compareElementsOfTwoForms(Form compareForm, Form
       formToCompareWith, String caller) {
2    String requester = caller;
3
4    Vector<FormElement> compareElements = compareForm.
       getElements();
5    Vector<FormElement> elementsToCompareWith =
       formToCompareWith.getElements();
6    FormElement compareElement;
7    FormElement elementToCompareWith;
8    int sameElementsCounter = 0;
9
10   for (int i = 0; i<compareElements.size(); i++) {
11     compareElement = compareElements.get(i);
12     elementToCompareWith = elementsToCompareWith.get(i);
13     if(compareElement.getID() == null && compareElement.
         getName() == null) {
14       if(elementToCompareWith.getID() == null &&
           elementToCompareWith.getName() == null) {
15         sameElementsCounter++;
16     } }
17     else if(compareElement.getID() == null && compareElement.
         getName() != null) {
18       if(elementToCompareWith.getID() == null &&
           elementToCompareWith.getName() != null) {
19         if(compareElement.getName().equals(
             elementToCompareWith.getName())) {
```

```
20              sameElementsCounter++;
21      } } }
22      else if(compareElement.getID() != null && compareElement.
            getName() == null) {
23          if(elementToCompareWith.getID() != null &&
                elementToCompareWith.getName() == null) {
24              if(compareElement.getID().equals(elementToCompareWith.
                    getID())) {
25                  sameElementsCounter++;
26      } } }
27      else if(compareElement.getID() != null && compareElement.
            getName() != null) {
28          if(elementToCompareWith.getID() != null &&
                elementToCompareWith.getName() != null) {
29              if(compareElement.getID().equals(elementToCompareWith.
                    getID()) && compareElement.getName().equals(
                    elementToCompareWith.getName())) {
30                  sameElementsCounter++;
31      } } }
32    }
33    if (compareElements.size() == sameElementsCounter) {
34      int score = compareAttributesOfTwoForms(compareForm,
            formToCompareWith, requester);
35      return score;
36    }
37    else {
38      // Vergleich abbrechen, beide Forms sind nicht gleich
39      return 1;
40  } }
```

Wurden zwei Formulare gefunden, die auf Elementebene die selben Inhalte aufweisen, erfolgt der dritte und letzte Vergleich. Dabei werden die beiden verglichenen Formulare auf ihre Gemeinsamkeiten bzgl. ihrer Attribute ID, Name und Action hin untersucht. Dieser Vergleich wird nur dann angestellt, wenn bereits durch die beiden vorherigen Vergleiche feststeht, dass die beiden Formulare gleich viele Elemente besitzen und diese darüber hinaus auch noch in ihren Attributen ID und Name übereinstimmen. Jedes als gleich identifizierte Formulare-Paar resultiert abhängig davon wofür der Vergleich genutzt wurde in unterschiedlichen Aktionen. Wird ein seiteninterner Vergleich angestellt so ist das Ziel festzustellen ob in einer Webseite mehrere Formulare existieren die als gleich zu betrachten sind. Dass Formulare der selben Webseite miteinander verglichen werden zeigt der Parameter *caller* an, der in diesem Fall mit dem Wert *"internal"* befüllt ist. Werden also zwei Formulare als gleich identifiziert wird bei beiden die Variable *multiple* auf *true* gesetzt. Dadurch werden die beiden Formulare als mehrfach vorkommend identifiziert und bei weiteren Vergleichen nicht mehr mit einbezogen, siehe Zeilen 7 bzw. 10 in Listing 4.1. Dies resultiert in bedeutend weniger Vergleichsoperationen, da

ein einmal als mehrfach vorkommend gekennzeichnetes Formular nicht erneut überprüft wird.

In Zeile 34 des Listings 4.2 wird die Methode *compareAttributesOfTwoForms(Form compareForm, Form formToCompareWith, String caller)* mit den beiden zu vergleichenden Formularen und dem String requester - sein Wert ist "internal" - aufgerufen. Dies geschieht nur, wenn bereits fest steht, dass die beiden übergebenen Formulare einander in ihren enthaltenen Elementen entsprechen. Die Überprüfung der einzelnen Formular-Attribute ist nicht besonders komplex, jedoch sehr umfangreich und wird deshalb nicht anhand eines Codeausschnitts erklärt. Im Prinzip bestehen die Überprüfungen aus einer Reihe von *if* und *else if* Verschachtelungen die die einzelnen Attribute der Formulare miteinander vergleichen. Der Rückgabewert der Vergleiche ist im Fall des Seiten-internen Vergleichs immer "-1" da die Variable caller den Wert "internal" hat und damit nur die *multiple* Variablen der verglichenen Formulare manipuliert wird. Hat sie "external" als Wert werden für jedes als gleich identifizierte Formular die ersten Punkte zum Gleichheits-Score hinzugefügt, dies ist jedoch nur beim Vergleich von Formularen aus unterschiedlichen Webseiten von Relevanz und ist Teil eines späteren Abschnitts. Hier hingegen werden noch keine Punkte vergeben, da es sich ja noch um den seiteninternen Vergleich zum finden mehrfach auftretender Formulare handelt.

Da Formulare die komplexesten Objekte sind die in einen Fingerabdruck mit einfließen ist ihr Bestandteil am Gesamtumfang des Programmcodes nicht unwesentlich. Die folgenden Aspekte einer Webseite sind bedeutend weniger umfangreich zu bearbeiten wie auch in den beigefügten Codeausschnitten ersichtlich ist.

4.1.3 HTML-Tags mit ID-Attribut

Im nächsten Schritt werden die in einer Webseite enthaltenen HTML-Konstrukte die ein "ID"-Attribut aufweisen zur Erstellung des Fingerabdrucks berücksichtigt.

Die dafür zuständige Methode *getIDs(List<Node> nodes)* wird mit einem *List* Objekt aufgerufen, das den gesamten HTML-Code der zu analysierenden Webseite in DOM-Form enthält und dadurch mittels XPath-Ausdrücken bearbeitet werden kann. Der XPath-Ausdruck *.//*[@id]/@id* liefert die Werte aller in der Seite enthaltenen ID-Attribute beliebiger HTML-Tags. Alle diese ID-Strings werden in einem java-Vector gespeichert und Duplikate sowie null-Werte werden ausgeschieden bevor der nächste Schritt folgt. Nach dieser Bereinigung werden alle verbliebenen IDs herangezogen um herauszufinden zu welchem HTML-Tag sie ursprünglich gehört haben. Das Ergebnis des XPath-Ausdrucks *.//*[@id]/@id* ist ein Objekt vom Typ *List* bestehend aus *org.w3c.dom.Attr* Objekten, das die String-Inhalte aller in der Seite enthaltenen ID-Attribute, unabhängig vom Namen des HTML-Knotens zu dem sie gehören, enthält.

Um zu jedem Attribut den zugehörigen HTML-Tag zu finden könnte der XPath-Ausdruck *name(/descendant::*[@id="x"][1])* verwenden werden, wobei das "x" für einen der zuvor gefundenen und gespeicherten ID-Strings steht. Der Ausdruck findet im HTML-Code den ersten Tag, dessen ID-Attribut mit "x" übereinstimmt und

liefert den Namen dieses Tags als Ergebnis. Damit auch bei mehrfach vorkommenden ID-Attributen immer das in der Seite als erstes vorkommende und bereits zuvor gespeicherte gefunden wird, wurde am Ende des XPath-Ausdrucks "[1]" angefügt. Dadurch wird sichergestellt, dass der von oben nach unten gesehen zu aller erst vorkommende HTML-Tag gefunden wird, der als ID "x" enthält.

Der Nachteil dieser Methode ist jedoch die Tatsache, dass ein XPath-Ausdruck wie */descendant::*[@id="x"][1]* sehr rechen- und damit auch zeitintensiv ist [24]. Das kommt insbesondere dann zum Tragen wenn sehr viele ID-Attribute in einer Webseite enthalten sind (siehe Tabelle 3.4). Aus diesem Grund wurde die Möglichkeit genutzt Methoden des *Attr* Objekts heranzuziehen mit Hilfe derer es möglich ist auf das die ID tragende Element zu schließen, bzw. den Namen dieses Elements festzustellen. Listing 4.3 zeigt wie dies im Programmcode umgesetzt wurde.

Nachdem der gesamte Inhalt der zu untersuchenden Webseite in einem *Node* Objekt gespeichert ist wird ein Objekt vom Typ *IDContainer* angelegt das den Rückgabewert der Methode darstellt. Bevor dieser IDContainer mit den korrekten Inhalten befüllt werden kann müssen alle leeren (Zeilen 9 bis 13) und mehrfach vorkommenden ID-Werte (Zeilen 17 bis 21) identifiziert und entfernt werden. Danach wird für jede verbleibende ID der dazugehörige HTML-Tag bestimmt und gemeinsam werden sie dem Rückgabeobjekt *IDContainer* als Objekt vom Typ *ID* hinzugefügt (Zeile 25 bzw. 26).

Listing 4.3: getIDs

```
1  public IDContainer getIDs(List<Node> nodes) {
2    Node node = nodes.get(0);
3    IDContainer IDContainer = new IDContainer();
4    try {
5      List<Node> IDNodes;
6      IDNodes = DOMHelper.XPath.evaluateXPath(node, "//*[@id]/
         @id");
7      List<Node> noNullIDs = IDNodes;
8      // Auf leere IDs prüfen, diese entfernen.
9      for(int i = 0; i<IDNodes.size(); i++) {
10       if(IDNodes.get(i).getTextContent().equals("")) {
11         noNullIDs.remove(i);
12         i--;
13     } }
14     List<Node> noDuplicateIDs = noNullIDs;
15     // Auf Duplikate prüfen, diese entfernen.
16     for(int i = 0; i<noNullIDs.size(); i++) {
17       for(int j = i+1; j<noDuplicateIDs.size(); j++) {
18         if(noNullIDs.get(i).getTextContent().equals(
             noDuplicateIDs.get(j).getTextContent())) {
19           noDuplicateIDs.remove(j);
20           j--;
21       } }
```

```
22          // Ergebnis von //*[@id]/@id ist eine Liste von
                Attributen.
23          // Auf Attr casten um den Namen des tragenden Tags zu
                finden.
24          Attr a = (Attr) noDuplicateIDs.get(i);
25          ID id = new ID(a.getOwnerElement().getTagName(),
                noDuplicateIDs.get(i).getTextContent());
26          IDContainer.addID(id);
27        }
28        return IDContainer;
29      }
30      catch (XPathSyntaxException e) {
31        e.printStackTrace();
32      }
33    return IDContainer;
34    }
```

Damit sind alle ID-Attribute einer Webseite gespeichert, Duplikate und leere IDs entfernt und zu jeder verbliebenen ID der Name des HTML-Elements zu dem das Attribut gehört gefunden und gespeichert. Die Behandlung der ID-Attribute und deren Vorbereitung zum Vergleich von Fingerabdrücken ist somit abgeschlossen.

4.1.4 HTML-Teilbaum

Als letzte Komponente fließt ein Teil der in der Webseite enthaltenen HTML-Struktur mit ein. Dieser Teil besteht aus den beiden dem <body> folgenden Ebenen.

Realisiert wird dies im ersten Schritt mittels des XPath-Ausdrucks *./body/**, dessen Ergebnis ein Objekt vom Typ *java.Util.List* ergibt, in dem alle dem Knoten <body> direkt folgenden Knoten enthalten sind. Diese in der Liste enthaltenen Knoten sind alle vom Typ *org.w3c.dom.Node*, und enthalten dadurch alle relevanten Informationen, das heißt es sind keine weiteren XPath-Ausdrücke mehr notwendig um die folgenden Schritte durchzuführen. Um den Knoten die gewünschten Eigenschaften und Verarbeitungsmöglichkeiten zu geben wurde die Klasse *BodyNode* eingeführt, die zwei Klassenvariablen beinhaltet, also jedem *BodyNode* Objekt zwei Eigenschaften verleiht. Diese Eigenschaften sind ein Name und ein *java.Util.Vector* Objekt in dem die Kind-Elemente des jeweiligen Knotens gespeichert sind.

Ist also ein Kind des <body> Knotens als *BodyNode* Objekte gespeichert kann mittels einer for-Schleife für jeden dieser Kind-Knoten festgestellt werden, ob wiederum auch dieser Kinder hat. Dies geschieht mittels einer if-Abfrage (Zeile 12) auf die Methode *hasChildNodes()*, die auf ein *org.w3c.dom.Node* Objekt angewendet *true* liefert wenn ein Knoten zumindest einen Kind-Knoten aufweist, bzw. *false* falls keine weiteren Elemente im betreffenden Knoten enthalten sind. Liefert die Methode also bei einem Kind von <body> true wird mittels einer weiteren for-Schleife (Zeile 13) unter Zuhilfenahme

der Methoden *getChildNodes()* die alle Kinder des betroffenen Elements liefert und *getLength()* die Anzahl der enthaltenen Kind-Elemente ermittelt und jedes enthaltene Element seinem Elternknoten als Kind zugewiesen (Zeile 16). Dies geschieht über die *BodyNode* Methode *addChild(BodyNode)* die das übergebene *BodyNode* Objekt jenem *BodyNode* zuweist auf den sie angewendet wurde. Die Kinder jedes *BodyNodes* sind in einem zugehörigen *Vector* Objekt gespeichert und ermöglichen es dadurch die Struktur des HTML-Baumes nachzubilden. Den *BodyNodes* die keine Kinder enthalten wird ein leerer *Vector* zugewiesen (Zeile 19).

Nachdem ein Kind des <body>-Knotens behandelt ist wird es als *BodyNode* Objekt im *BodyNodeContainer* gespeichert (Zeile 21). Der *BodyNodeContainer* ist eine Klasse die dazu dient alle Objekte des Typs *BodyNode* einer Webseite gemeinsam zu speichern und wird mit allen gefundenen Kind-Knoten des HTML-Bodies befüllt, die wiederum auch ihre eigenen Kind-Knoten enthalten. Bevor dieser *BodyNodeContainer* als Ergebnis zurückgeliefert werden kann muss jedoch der Befehl *FingerprintBrowser.display.dispose();* (Zeile 24) ausgeführt werden, der die Verbindung zwischen der aufgerufenen Webseite und der aufrufenden Applikation trennt. Dies wird an genau dieser Stelle gemacht da danach keine weiteren Zugriffe auf den HTML-Code der Webseite mehr erfolgen. Das Objekt *FingerprintBrowser* ist eine Klasse die eine Instanz des Mozilla Webbrowsers (siehe auch Abschnitt 5.2) realisiert, über die die Kommunikation zwischen Applikation und WWW ermöglicht wird und die zum Beispiel feststellt wann eine Webseite vollständig geladen ist um sie bearbeiten zu können.

Listing 4.4: getBodyNodes

```
1  public BodyNodeContainer getBodyNodes(List<Node> nodes) {
2    Node node = nodes.get(0);
3    List<Node> bodyNodes;
4    BodyNodeContainer bnc = new BodyNodeContainer();
5    try {
6      bodyNodes = DOMHelper.XPath.evaluateXPath(node, "./body/*"
           );
7      // Leerer Vector für Knoten in 3. Ebene (deren Kind-Knoten
           bleiben unberücksichtigt)
8      Vector emptyVector = new Vector(0);
9      for (int i=0; i<bodyNodes.size(); i++) {
10       Node n = bodyNodes.get(i);
11       BodyNode bodyNode = new BodyNode(n.getNodeName());
12       if (n.hasChildNodes()) {
13         for (int j=0; j<n.getChildNodes().getLength(); j++) {
14           String nameOfNode = n.getChildNodes().item(j).
               getNodeName();
15           BodyNode child = new BodyNode(nameOfNode);
16           bodyNode.addChild(child);
17       } }
18       else {
19         bodyNode.setChildren(emptyVector);
```

```
20          }
21        bnc.addNode(bodyNode);
22        }
23        // Display Objekt verwerfen.
24        FingerprintBrowser.display.dispose();
25        // Korrektes Ergebnis.
26        return bnc;
27      }
28      catch (XPathSyntaxException e) {
29       e.printStackTrace();
30      }
31      return bnc;
32  } }
```

Nachdem nun auch der HTML-Body-Teilbaum aufbereitet ist um in einen Fingerabdruck mit einzufließen sind alle Schritte zur Erstellung eines Fingerabdrucks abgeschlossen. Die folgenden Abschnitte befassen sich nun damit wie zwei Fingerabdrücke miteinander verglichen werden und wie festgestellt wird welchem bereits existierenden Fingerabdruck der zu vergleichende am ähnlichsten ist.

4.2 Vergleich von Fingerabdrücken

Grundsätzlich basiert der Vergleich auf einem Punktesystem, d.h. auf jeder Vergleichsebene (URL, HTML-Forms, HTML-IDs und HTML-Body-Teilbaum) werden Punkte für Gemeinsamkeiten vergeben. Derjenige Fingerabdruck aus dem Pool der Fingerabdrücke bekannter Webseiten der beim Vergleich die höchste Punkteanzahl erreicht wird als der gesuchten Seite entsprechend interpretiert. Auf das Punktesystem wird zu Beginn dieses Kapitels eingegangen bevor die Vergleichsalgorithmen der einzelnen Komponenten beschrieben werden.

4.2.1 Punktesystem

Die aus den vier genannten Ebenen bestehende Vergleichslogik braucht ein Mittel um die Ähnlichkeit zweier Fingerabdrücke zueinander zum Ausdruck zu bringen. Dies ist im vorliegenden Fall mittels eines Punktesystems umgesetzt, das für alle vier zu vergleichenden Webseiten-Komponenten auf unterschiedliche Weise Punkte für bestehende Ähnlich- bzw. Gleichheit vergibt. Hier wird in Tabelle 4.1 kurz zusammengefasst wofür wieviele Punkte vergeben werden; warum diese Punktezahlen gewählt wurden wird daran anschließend beschrieben.

Tabelle 4.1: Punktevergabe

Komponente	**Punkte**	**Kriterium**
URL	1	Pro übereinstimmendem Zeichen.
HTML-Form	5	Für ein Formular mit gleicher Elementanzahl, gleichem Namen und ID aller Elemente, und 2 gleichen von 3 Form-Attributen.
HTML-Tags mit ID	1	Tagname und ID-Wert gleich.
HTML-Body-Teilbaum	25	Völlige strukturelle Übereinstimmung.

Wenn zwei URLs miteinander verglichen werden, so werden sie von links nach rechts, Buchstabe für Buchstabe verglichen. Sobald ein Zeichen auftritt, das an der momentan untersuchten Stelle nicht in beiden URLs zu finden ist wird der Vergleich beendet. Die Anzahl an gleichen Buchstaben bis zum Abbruch des Vergleichs ergibt die zu vergebende Anzahl an Punkten. Ist es jedoch der Fall dass zwei URLs einander vollständig entsprechen, was einerseits sehr aussagekräftig, andererseits jedoch außer bei den Suchseiten wegen den vielen Parametern die in den URLs der anderen beiden Seitenkategorien enthalten sind sehr unwahrscheinlich ist, werden 75 Punkte vergeben. 75 wurde gewählt, da dies ein Wert ist der größer ist als die Anzahl enthaltener Character in jeder der Suchseiten der verwendeten Quell-Webseiten. Dadurch werden bei absoluter Gleichheit zweier URLs, was wie erwähnt nur bei Suchseiten zu erwarten ist, mehr Punkte vergeben als bei Anwendung der eigentlichen URL-Vergleichstechnik erzielt werden können. Dadurch soll eine gewisse Lenkung des Vergleichsergebnisses in die richtige Richtung erfolgen, denn es wird davon ausgegangen, dass zwei Webseiten mit der exakt selben URL auch als gleich zu betrachten sind.

Werden zwei Fingerabdrücke miteinander verglichen und findet sich in diesen beiden Objekten jeweils ein Formular das die gleiche Anzahl an enthaltenen Elementen aufweist, diese Elemente in ihren Attributen *Name* und *ID* die selben Werte beinhalten und auch noch in zwei der drei festgehaltenen Formular-Attribute *Name*, *ID* und *Action* die selben Werte enthalten sind werden sie als einander entsprechend eingestuft. Wird in einem zu untersuchenden Fingerabdruck ein diesen Kriterien entsprechendes Formular gefunden, so werden dem Fingerabdruck dafür fünf Punkte angerechnet. Zu Beginn der Entwicklung des Punktesystems wurde mit 25 Punkten für jedes gefundene Formular gerechnet, was jedoch in einzelnen Fällen zu sehr hohen Punktwerten geführt hat. Um die Formulare nicht zu einem dominierenden Faktor in der Punktevergabe werden zu lassen wurde die Höhe der für gleiche Formulare zu vergebenen Punkte nach einigen Versuchen mit unterschiedlichen Werten auf fünf festgelegt.

Der Vergleich zweier Fingerabdrücke auf Ebene der HTML-Tags mit ID-Attribut resultiert in einem zusätzlichen Punkt für den verglichenen Fingerabdruck wenn er ein ID-Objekt enthält, das mit selbem Namen und selbem ID-Inhalt auch im zu vergleichenden Fingerabdruck vorhanden ist. Verschiedene Webseiten (siehe 3.4) weisen große Zahlen enthaltener HTML-Tags mit ID-Attribut auf. Dadurch wird die Gesamtsumme

der vergebenen Punkte eines Vergleichs oft durch sie dominiert. Da diese IDs jedoch Website- sowie Kategoriespezifisch sind ist eine gewisse Dominanz durch sie kein Nachteil. Wie auch bei den Formularen wurden in einem früheren Stadium der Entwicklung mehr Punkte für Übereinstimmungen auf dieser Ebene vergeben als letztlich festgelegt wurde. Nach einigen Tests wurde diese Zahl von drei auf eins verringert, denn mehr als einen Punkt zu vergeben würde die Dominanz im Relation zu den anderen Vergleichsebenen zu sehr erhöhen und die Verwendung der anderen Bestandteile im Vergleich nicht mehr rechtfertigen.

Weist ein Fingerabdruck ein strukturell absolut gleich aufgebautes HTML-Body-Teilbaum Objekt auf wie es im zu findenden Fingerabdruck vorhanden ist, so werden diesem Fingerabdruck 25 Punkte dafür gutgeschrieben. Nachdem, wie auch in den Tabellen des Abschnitts 6 ersichtlich, der HTML-Body-Teilbaum häufig nicht gleich ist, obwohl die beiden verglichenen Webseiten einander entsprechen wurde ein Wert verwendet der keine entscheidende Auswirkung auf das Vergleichsergebnis hat. Mit 25 Punkten kann er jedoch in Kombination mit den anderen Vergleichsebenen seinen Einfluss einbringen ohne ein entscheidender Faktor zu sein.

Aus diesen vier Einzelwerten wird am Ende des Vergleichs eine Summe gebildet. Derjenige Fingerabdruck aus dem Pool der bekannten Webseiten der dann die höchste Gesamtpunktezahl aufweist wird der zu findenden Webseite als am ähnlichsten eingestuft und als "Gewinner" des Vergleichs ausgegeben.

4.2.2 URLs

Beim URL-Vergleich werden die beiden URLs der zu vergleichenden Webseiten von links nach rechts Zeichen für Zeichen miteinander verglichen und es wird für jedes übereinstimmende Zeichen ein Punkt vergeben. Der Vergleich bricht beim ersten nicht übereinstimmenden Zeichen ab; dadurch ergibt sich für jede mittels Fingerabdruck abgespeicherte Webseite ein eigener Wert, der bei der Seite die der gesuchten entspricht am höchsten sein sollte. Da damit jedoch nicht immer eine eindeutige Zuordnung durchgeführt werden kann, bzw. es auch möglich ist dass mehrere der hinterlegten Fingerabdrücke auf URL-Ebene die gleiche Punkteanzahl erreichen, erfolgen daraufhin die Vergleiche auf den drei weitere Ebenen um eine eindeutige Zuordnung ermöglichen zu können.

Die zu vergleichenden Fingerabdrücke wurden der Methode *compare()* aus Listing 4.6 über den Konstruktor der beinhaltenden Klasse übergeben. In den Zeilen zwei und drei werden die URLs der betreffenden Fingerabdrücke ausgelesen und in den beiden folgenden if-Abfragen auf eine Länge von 200 Zeichen beschränkt. Sie nach 200 Zeichen abzuschneiden beruht auf der Annahme dass URLs großer Länge und mit vielen enthaltenden Parametern sich aufgrund der variablen Inhalte nach bereits wenigen Zeichen nicht mehr gleichen und so Rechenzeit gespart werden kann. Wie in den Tabellen des Kapitels 6 ersichtlich gab es keine zwei URLs die mehr als 110 gleiche Zeichenfolgen aufzuweisen hatten was diese Beschränkung rechtfertigt.

In den Zeilen 17 - 20 des folgenden Codeausschnitts 4.5 wird überprüft ob die beiden URLs absolut gleich sind wodurch der Vergleich Zeichen für Zeichen nicht durchgeführt wird, sondern standardmäßig 75 Punkte vergeben werden. In den Zeilen 26 bis 37 wird die Anzahl der gleichen Character ermittelt und in Zeile 44 wird diese Anzahl dem verglichenen Fingerabdruck als Punkte aus dem URL-Vergleich hinzugefügt.

Listing 4.5: compareURLs

```
1  public void compare() {
2    String compareURL = compareFp.getUrl();
3    String URLToCompareWith = fpToCompareWith.getUrl();
4
5    int compareURLLength = compareURL.length();
6    int URLToCompareWithLength = URLToCompareWith.length();
7    if(compareURLLength > 200) {
8      compareURL = compareURL.substring(0, 200);
9    }
10   if(URLToCompareWithLength > 200) {
11     URLToCompareWith = URLToCompareWith.substring(0, 200);
12   }
13   StringBuffer commonStringBuffer = new StringBuffer();
14   String commonString = "";
15
16   // Test ob beide URLs völlig gleich sind.
17   if (compareURL.equals(URLToCompareWith)) {
18     // Gesamtscore des Fingerabdrucks um 75 Punkte erhöhen
19     fpToCompareWith.addScore(75);
20     fpToCompareWith.setURLScore(75);
21   }
22   else {
23     boolean stop = false;
24     try {
25       // Zeichen für Zeichen vergleichen
26       for(int i = 0; stop != true; i++) {
27         if(i<compareURL.length() && i<URLToCompareWith.length
               ()) {
28           if(compareURL.charAt(i) == URLToCompareWith.charAt(i
                 )) {
29             Character c = compareURL.charAt(i);
30             commonStringBuffer.append(c);
31           }
32           else {
33             stop = true;
34         } }
35         else {
36           stop = true;
```

```
37      } } }
38      catch (Exception e) {
39        System.out.println("CompareURLs exception: " + e);
40        System.exit(1);
41      }
42      commonString = commonStringBuffer.toString();
43      // Anzahl gleicher Characters als Punkte vergeben.
44      fpToCompareWith.addScore(commonString.length());
45      fpToCompareWith.setURLScore(commonString.length());
46  } }
```

4.2.3 Formulare

Nach dem Vergleich der URLs der Fingerabdrücke folgt der Vergleich der in den einzelnen Seiten enthaltenen Formulare. Dieser erfolgt zwischen allen enthaltenen Formularen und lässt sich grundsätzlich in zwei Teilbereiche aufteilen.

1. Vergleich der enthaltenen Elemente
2. Vergleich der Formular-Attribute

Zwei Formulare werden nur dann weiter miteinander verglichen wenn sie die selbe Anzahl enthaltener Elemente aufweisen. Durch Entfernung doppelter Einträge bei der Erstellung des Fingerabdrucks wurde bereits gewährleistet dass in jedem Formular nur mehr einzeln vorkommende Elemente enthalten sind.

Enthalten zwei Formulare die selbe Anzahl an Elementen werden alle diese einzeln und jedes mit jedem verglichen. Verglichen wird dabei, ob es für jedes Element des aktuellen Formulars aus der zu vergleichenden Webseite im zu diesem Zeitpunkt untersuchten Formular der verglichenen Webseite ein Element gibt, das in seinen Attributen Name und ID exakt die selben Inhalte aufweist. Diese Werte können vom Typ *String* sein oder aber *null*.

Wird beim Elementevergleich festgestellt, dass sowohl die Anzahl der enthaltenen Elemente als auch deren IDs und Namen einander entsprechen wird der Vergleich der Formular-Attribute eingeleitet. Dabei werden die drei Attribute ID, Name sowie Action berücksichtigt, die einen *String* oder auch *null* als Wert enthalten können. Stimmen zwei der drei Attribute überein so werden die beiden Formulare als gleich angesehen und der Vergleich ist damit abgeschlossen. Umgesetzt ist dieser Vergleich über die Methode des Listings 4.1 wo die private Methode *compareElementsOfTwoForms* aufgerufen wird, die wiederum die Methode *compareAttributesOfTwoForms* aufruft. Der Aufruf erfolgt diesmal jedoch mit dem Wert *"external"* im Parameter *caller* durchgeführt, wodurch nicht die *multiple* Variable von Forms manipuliert wird, sondern der Score des verglichenen Fingerabdrucks um fünf erhöht wird wenn ein dem gesuchten Formular als gleich einzustufendes gefunden wurde.

4.2.4 HTML-Tags mit ID-Attribut

Jeder HTML-Tag der ein ID-Attribut aufweist und der mit Name und ID-Inhalt in einem anderen Fingerabdruck wiedergefunden wird resultiert in einem zusätzlichen Punkt im Gesamtscore des untersuchten Fingerabdrucks. Der ID Vergleich ist in einigen Fällen sehr einflussreich, insbesondere bei Webseiten die eine große Anzahl an HTML-Tags mit ID-Attribut aufweisen (vgl. Tabelle 3.4), beispielsweise die Formularseite von Expedia oder die Übersichtsseite von Venere.

Nachdem der Vergleich alle (bis auf Duplikate und IDs die "null" enthalten) beinhalteten ID-Strings berücksichtigt ist es irrelevant, ob es IDs gibt, die in allen Seitenkategorien vorkommen und dadurch auch immer gefunden werden. Ebenso unerheblich ist die Tatsache, dass von den dargestellten Inhalten abhängige IDs wie solche die Identifikationsnummern oder -strings von Hotels, Zimmern oder Angeboten enthalten vorhanden sein können, die, wenn sich die Inhalte der Webseite geändert haben (beispielsweise wird eine andere Stadt abgefragt), nicht mehr gefunden werden können. Die seitenspezifischen Inhalte werden immer ebenfalls gefunden und die Identifikation ist nur über diese Elemente möglich. Daher können variable und auch in allen Kategorien vorkommende Tags und ihre IDs die Vergleichsergebnisse nicht beeinflussen.

Im Programmcode ist der Vergleich der HTML-Tags und ihrer ID Attribute wie in Listing 4.6 ersichtlich umgesetzt. Da die zu vergebenden Punkte den betreffenden Fingerabdrücken direkt zugewiesen werden hat auch diese Methode keine Werte die sie an die aufrufende Funktion zurück liefert. Aus den an die Methode übergebenen Fingerabdrücken werden die *IDContainer*-Objekte ausgelesen und über die zwei *for*-Schleifen der Zeilen 7 bis 12 vollständig miteinander in Beziehung gesetzt. In Zeile 9 wird verglichen ob die beiden zu diesem Zeitpunkt untersuchten *ID*-Objekte den selben Tagnamen und ID-Attribut-Inhalt aufweisen. Ist dies der Fall wird das Objekt *score* um einen Zähler erhöht, und der Wert dieses Objekts wird in weiterer Folge dem Gesamtscore des verglichenen Fingerabdrucks als Ergebnis des ID-Vergleichs hinzugefügt (Zeile 13).

Listing 4.6: compareIDs

```
1  public void compare(Fingerprint compareFp, Fingerprint
       fpToCompareWith) {
2    this.compareFp = compareFp;
3    IDContainer compareIDs = compareFp.getIDContainer();
4    this.fpToCompareWith = fpToCompareWith;
5    IDContainer IDsToCompareWith = fpToCompareWith.
       getIDContainer();
6    int score = 0;
7    for (int i=0; i<compareIDs.getNumberOfIDs(); i++) {
8      for (int j=0; j<IDsToCompareWith.getNumberOfIDs(); j++) {
9        if(compareIDs.getIDs().get(i).getTagname().equals(
           IDsToCompareWith.getIDs().get(j).getTagname()) &&
           compareIDs.getIDs().get(i).getId().equals(
           IDsToCompareWith.getIDs().get(j).getId())) {
```

```
10            // Tagname und ID gleich, + 1 Punkt
11            score += 1;
12    } } }
13    fpToCompareWith.addScore(score);
14    fpToCompareWith.setIDScore(score);
15 }
```

4.2.5 HTML-Teilbaum

Der Vergleich der HTML-Teilbaum Strukturen ist ganz ähnlich aufgebaut wie die vorherigen Vergleichsmechanismen. Bevor zwei Fingerabdrücke verglichen werden können wird ein neues Objekt vom Typ *CompareBodyNodes* angelegt, das es ermöglicht die Methode *compare(Fingerprint, Fingerprint)* aufzurufen. An diese Methode werden die zwei zu vergleichenden Fingerabdrücke übergeben und anhand ihrer enthaltenen *BodyNodeContainer* Objekte verglichen. Zwei Fingerabdrücke werden nur dann als gleich betrachtet, wenn sie in ihren HTML-Strukturen zwei Ebenen vom <body> abwärts völlig übereinstimmen. Als erstes wird also verglichen ob die <body> Elemente der beiden Fingerabdrücke gleich viele Kindelemente aufweisen (Listing 4.7, Zeile 6). Nur wenn dies der Fall ist werden auch weitere Vergleiche angestellt. Andernfalls werden dem gerade verglichen Fingerabdruck keine weiteren Punkte zum Gesamtscore hinzugezählt (Zeile 35).

Nachdem also festgestellt ist, dass beide Fingerabdrücke auf Ebene der Body-Kinder gleich viele Elemente aufweisen, wird ein detaillierterer Vergleich durchgeführt. Mittels einer for-Schleife (Zeile 8) werden nacheinander alle direkten Nachfolger des Body-Elements herangezogen und es wird überprüft, ob das gerade aktuelle *BodyNode* Objekt im zu vergleichenden Fingerabdruck ebenfalls vorhanden ist (Zeile 9). Diese Überprüfung bezieht sich auf den Namen des Elements, die Anzahl der enthaltenen Kind-Knoten sowie die Position im *BodyNodeContainer*. Nur wenn die beiden zu vergleichenden *BodyNodes* in allen drei Punkten übereinstimmen wird der Vergleich fortgesetzt, andernfalls wird der Vergleich abgebrochen, die beiden *BodyNodeContainer* und damit auch die beiden Fingerabdrücke in denen sie gespeichert sind, werden als nicht gleich definiert und keine Punkte vergeben.

Stimmen die beiden zu diesem Zeitpunkt zu vergleichenden *BodyNodes* der ersten Kind-Ebene von <body> jedoch in den drei zuvor genannten Details überein kommt es zum nächsten Vergleich. Bei diesem letzten Vergleich wird festgestellt, ob alle in den beiden behandelten *BodyNodes* enthaltenen Subknoten, also Kindeskinder (oder Enkel) von <body> in der selben Reihenfolge und auch mit dem selben Namen vorhanden sind (Zeile 14). Ist dies bei allen Knoten der Fall so werden die beiden *BodyNodeContainer* der verglichenen Fingerabdrücke und damit auch die Fingerabdrücke selbst als gleich angesehen. Für Gleichheit auf Ebene der *BodyNodes* werden zum Gesamtscore der gerade verglichenen Webseite 25 Punkte hinzugefügt.

Die Umsetzung auf Programmierebene ist in Listing 4.7 festgehalten. Nachdem die

BodyNode-Objekte der zu vergleichenden Fingerabdrücke ausgelesen sind wird in Zeile 5 ein Objekt namens *unequal* und vom Typ *boolean* angelegt, das dazu genützt wird festzuhalten ob zwei verglichene *BodyNode*-Objekte einander entsprechen oder nicht und schlussendlich bestimmt, ob Punkte vergeben werden oder nicht (Zeilen 30 bis 37). Da diese Methode ein wenig unübersichtlich ist wurden hier mehrere Kommentare eingefügt die zum leichteren Verständnis beitragen sollen.

Listing 4.7: compareBodyNodes

```
1  public void compare(Fingerprint compareFp, Fingerprint
       fpToCompareWith) {
2    Vector<BodyNode> compareBodyNodes = compareFp.
       getBodyNodeContainer().getBodyNodes();
3    Vector<BodyNode> bodyNodesToCompareWith = fpToCompareWith.
       getBodyNodeContainer().getBodyNodes();
4    // not equal flag
5    boolean unequal = false;
6    if (compareBodyNodes.size() == bodyNodesToCompareWith.size()
       ) {
7    // Beide bodies gleiche Anzahl an Subknoten.
8      for (int i=0; i<compareBodyNodes.size(); i++) {
9        if (compareBodyNodes.get(i).getName().equals(
           bodyNodesToCompareWith.get(i).getName()) &&
10           compareBodyNodes.get(i).getNumberOfChildren() ==
               bodyNodesToCompareWith.get(i).getNumberOfChildren
               ())
11       {
12         // Dzt. verglichene Kinder der beiden bodies haben
               gleich viele Kinder und den selben namen.
13         for (int j=0; j<compareBodyNodes.get(i).getChildren().
             size(); j++) {
14           if (compareBodyNodes.get(i).getChildren().get(j).
               getName().equals(bodyNodesToCompareWith.get(i).
               getChildren().get(j).getName()))
15           {
16             // Dzt. verglichene Kinder haben selben namen,
                   unequal bleibt false.
17           }
18           else {
19             // Dzt. verglichene Kinder haben verschiedene
                   namen, unequal wird true gesetzt.
20           unequal = true;
21       } } }
22       else {
23       // Dzt. verglichene Kinder der beiden bodies haben nicht
             gleich viele Kinder und/oder selben namen; nicht
```

```
            vergleichen.
24          unequal = true;
25      } } }
26      else {
27        // Bodies unterschiedliche enthaltene knotenanzahl, nicht
              vergleichen.
28        unequal = true;
29      }
30      if (unequal == false) {
31        fpToCompareWith.addScore(25);
32        fpToCompareWith.setBodyNodeScore(25);
33      }
34      else {
35        fpToCompareWith.addScore(0);
36        fpToCompareWith.setBodyNodeScore(0);
37  } }
```

Kapitel 5

Prototypische Implementierung

Die Entwicklung der beschriebenen Applikation ist ein essenzieller Bestandteil dieser Arbeit. Ungefähr die Hälfte der gesamten für die Erstellung der Arbeit aufgewendeten Zeit wurde in die Programmierung investiert. Aufgrund des Umfanges des Programmcodes (18 Klassen mit etwa 3000 Zeilen Code) ist es nicht möglich den gesamten Programmcode im Anhang einzufügen; der Arbeit ist jedoch eine CD beigelegt auf der der gesamte Code zu finden ist. Einige der wichtigsten Klassen und Implementierungsentscheidungen sind in diesem Abschnitt beschrieben, neben den Hinweisen dafür wie die Applikation korrekt verwendet werden kann.

Vorweg ist grundsätzlich festzuhalten, dass es sich um eine Kommandozeilen-basierte Applikation handelt. Als Name der Applikation wurde "Fingerprintchecker" gewählt. Die Nutzung und Steuerung der Applikation erfolgt durch Eingabe bestimmter "Schlüsselwörter" und Parameter über die Kommandozeile. Der Aufbau der Befehle ist einfach und immer gleich und setzt sich folgendermaßen zusammen:

java -jar Fingerprintchecker.jar {Schlüsselwort} [{Name} {URL}]

Der Ausdruck "java" startet die Klasse Main der Applikation und übergibt die Parameter Schlüsselwort, Name und URL zur Verarbeitung. Das Schlüsselwort muss einem der vorgegebenen Begriffe die zur Steuerung des Verhaltens des Programms vorgesehen sind entsprechen. Der Parameter Name muss gesetzt sein wenn ein Fingerabdruck zum Pool der bekannten Seiten hinzugefügt werden soll und wenn ein bestimmter Fingerabdruck aus dem Pool entfernt werden soll. Der letzte Parameter ist die URL die aufgerufen werden soll und zu der Webseite führt deren Fingerabdruck anzulegen oder zu vergleichen ist. Der folgende Abschnitt beschreibt die Interaktionsmöglichkeiten mit dem Programm näher.

5.1 Anwenwendungsvorgaben

Es sind insgesamt sieben Anwendungsmöglichkeiten der Applikation umgesetzt die die denkbaren Vorgehensweisen im Zusammenhang mit der Nutzung von Fingerabdrücken

und deren Wiedererkennung abdecken und ermöglichen. Die verwendbaren Schlüsselwörter sind in der folgenden Aufstellung in Erweiterter Backus-Naur-Form (EBNF) [34] zusammengefasst:

```
START = 'java -jar Fingerprintchecker.jar' INPUT ;
INPUT = 'info' |
        'list' |
        'add' NAME URL |
        'replace' NAME URL |
        'remove' NAME |
        'clear' |
        'find' URL ;
NAME = ? beliebige Zeichenfolge ? ;
URL  = ? korrekte URL ? ;
```

Grundsätzlich will der Benutzer, dass die Applikation dazu in der Lage ist in einem Pool von Webseiten diejenige wiederzufinden die der Webseite entspricht die er der Anwendung als Input übergibt. Um dies zu ermöglichen sind jedoch einige andere Anwendungsfälle von Nöten bevor man damit anfangen kann Webseiten zu vergleichen und wiederzufinden.

Der folgende Befehl gibt dem Benutzer einen Überblick über alle möglichen Eingabevarianten und deren Effekte.

java -jar Fingerprintchecker.jar info

Zu Beginn könnte man sich ansehen wie die im System vorhandenen Fingerabdrücke bezeichnet wurden und welche URLs sie repräsentieren.

java -jar Fingerprintchecker.jar list

Damit das Programm eine bestimmte Webseite in einer Menge von bekannten Webseiten finden kann muss zu aller erst die Gruppe der bekannten Webseiten an das System übergeben werden. Dazu dient das Schlüsselwort *add*, das dazu führt, dass der Fingerabdruck der durch die übergebenen URL repräsentierten Webseite in der Applikation persistent gespeichert wird. Um einen Fingerabdruck zum Pool der bekannten Websites hinzuzufügen muss die Anwendung mit dem Befehl

java -jar Fingerprintchecker.jar add [Name] [URL]

gestartet werden. Die übergebene URL wird aufgerufen, wenn die Webseite vollständig übermittelt wurde wird der Fingerabdruck für sie erstellt und zur Gruppe der bekannten Webseiten hinzugefügt. Auf diese Weise lässt sich der Fingerabdruck-Pool mit beliebig vielen Webseiten füllen.

Wird eine neue Webseite zum Pool hinzugefügt wird zuvor überprüft ob bereits ein Fingerabdruck mit dem angegebenen Namen enthalten ist und wenn ja der Benutzer darauf hingewiesen eindeutige Bezeichnungen zu verwenden. Hat man vor einen bereits bestehenden Fingerabdruck zu erneuern so muss man neben dem Stichwort ”replace” den im Pool bereits vorhandenen Namen des zu ersetzenden Fingerabdrucks angeben.

Der unter diesem Namen vorhandene Fingerabdruck wird verworfen und durch den aktuellen Fingerabdruck aufgrund der übergebenen URL ersetzt. Der Befehl

java -jar Fingerprintchecker.jar replace [Name] [URL]

führt dies aus und soll angewendet werden wenn eine Webseite einer Überarbeitung durch die Betreiber unterzogen wurde. Dies dient dazu den Referenzfingerabdruck wieder auf den neuesten Stand zu bringen und die neuen Inhalte der Webseite darin widerzuspiegeln damit ein Vergleich wieder korrekt durchgeführt werden kann.

Wird ein Fingerabdruck nicht mehr im Pool gewünscht kann er ganz einfach mittels

java -jar Fingerprintchecker.jar remove [Name]

entfernt werden. Auch die Möglichkeit den gesamten Pool zu leeren ist vorhanden, die Anweisung

java -jar Fingerprintchecker.jar clear

löscht alle persistent gespeicherten Fingerabdrücke. Die wichtigste Eingabe ist jedoch die mit der man eine URL an das System übergibt um diese Webseite in der Gruppe der gespeicherten Fingerabdrücke wiederzufinden. Mittels

java -jar Fingerprintchecker.jar find [URL]

wird das Programm so gestartet, dass zuerst von der Webseite der übergebenen URL ein Fingerabdruck erstellt wird der dann in weiterer Folge mit allen im Pool der bekannten Fingerabdrücke vorhandenen verglichen wird um den ihm ähnlichsten zu finden.

Diese Nutzungsvorgaben finden sich gemeinsam mit der Installationsanleitung auch in Anhang B.

5.2 Essenzielle Klassen

Insgesamt besteht die erstellte Applikation aus 18 Klassen die in Anhang A als Klassendiagramm dargestellt sind. Die Grafik befindet sich im Anhang, da die Abbildung sehr überladen und wenig übersichtlich ist. Im vorliegenden Abschnitt geben Abbildung 5.1 bis 5.5 einen Überblick über die gesamte Applikation und zeigen die enthaltenen Komponenten und deren Zusammenspiel anhand von UML-Package-Diagrammen.

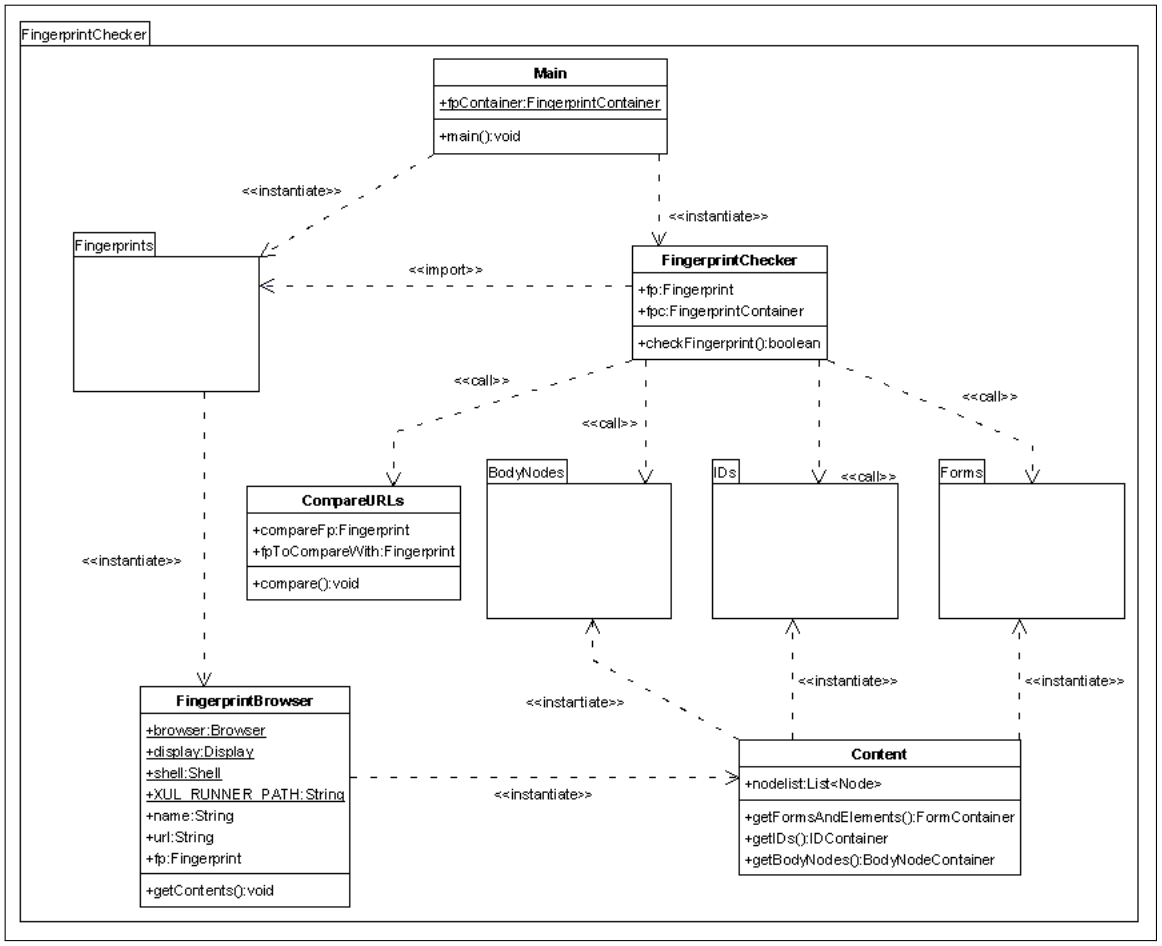


Abbildung 5.1: Fingerprintchecker-Paket

Die Applikation setzt zwei Hauptaufgaben um, einerseits die Erstellung von Fingerabdrücken, andererseits deren Vergleich. In der Klasse *Main* hat die Applikation ihren Ausgangspunkt und wurde eine URL zur Erstellung eines Fingerabdrucks übergeben, wird zu Beginn das Package *Fingerprints* angesprochen. Ein *Fingerprint* Objekt wird instanziiert und die Klasse *FingerprintBrowser* aufgerufen, die die Inhalte der aufzurufenden Webseite an die Applikation überträgt und zur Weiterverarbeitung aufbereitet. Über die Klasse *Content* werden danach die einzelnen Pakete aufgerufen die die jeweiligen Inhalte auslesen und die dementsprechenden Objekte anlegen die gemeinsam den Fingerabdruck bilden. In der Klasse *FingerprintBrowser* wird der HTML-Code der aufgerufenen Webseite in ein DOM-Dokument umgewandelt um die in vorherigen Abschnitten erwähnten XPath Ausdrücke überhaupt anwenden zu können. Diese wichtige Funktionalität wird über die Mozilla Programmbibiliothek *MozillaInterfaces.jar* ermöglicht, die ein Teil des XULRunner Projekts ist, das auf [1] beschrieben und auf [2] zum Download bereitgestellt ist. Das daraus verwendete Java Archive findet sich im Ordner *xulrunner/sdk/lib*.

[1] https://developer.mozilla.org/en/XULRunner
[2] http://ftp.mozilla.org/pub/mozilla.org/xulrunner/nightly/latest-trunk/

Sollen zwei Fingerabdrücke miteinander verglichen werden, muss zuerst von der übergebenen URL ein Fingerabdruck wie direkt zuvor beschrieben erstellt werden. Main ruft im Anschluss daran die Klasse *FingerprintChecker* mit diesem Fingerabdruck auf und holt aus der Klasse *FingerprintContainer* des Paktes *Fingerprints* alle dort hinterlegten Fingerabdrücke. Um den wiederzufindenden Fingerabdruck mit jenen aus dem Pool in Beziehung zu setzen werden nacheinander die vier Vergleichsmethoden *CompareURL*, *CompareForms*, *CompareIDs* und *CompareBodyNodes* aufgerufen. Da jede Webseite nur genau eine URL hat benötigt die Klasse *CompareURL* keine Hilfsklasse in der, wie bei den anderen drei HTML-Komponenten, die Objekte gleichen Typs gesammelt gespeichert sind.

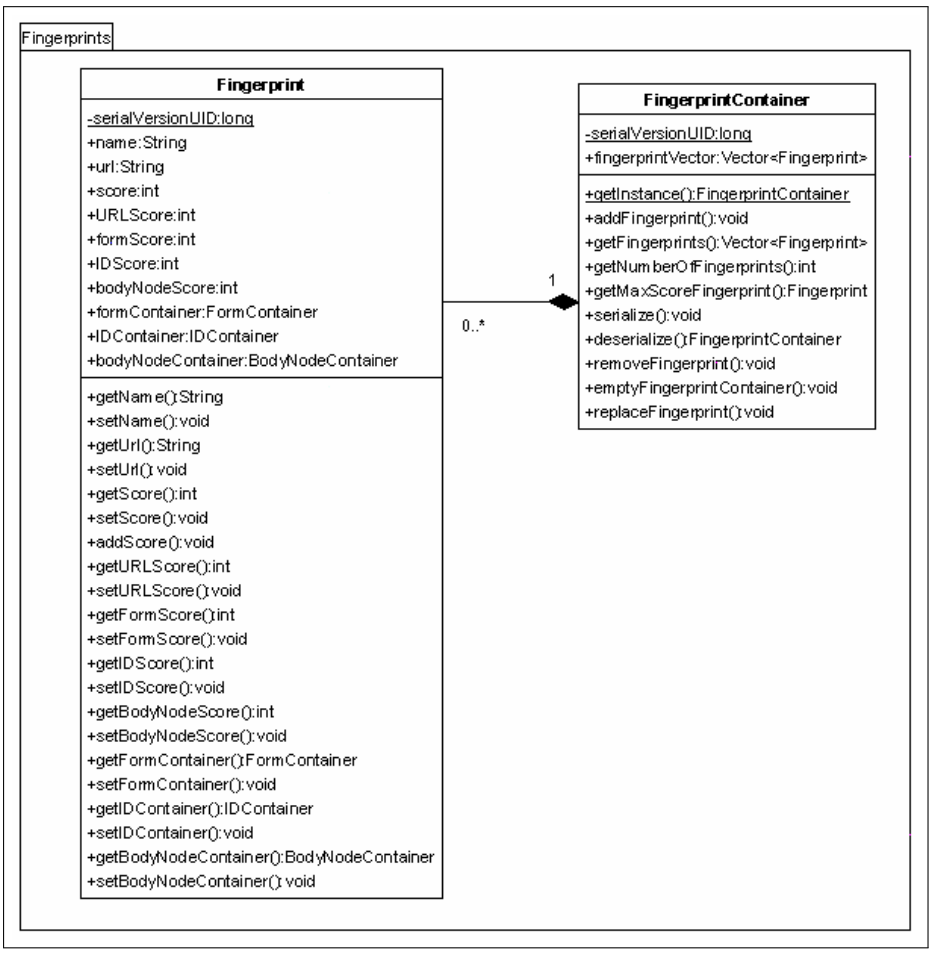


Abbildung 5.2: Fingerprints-Paket

Das Paket *Fingerprints*, Abb. 5.2 enthält die Klassen *Fingerprint* sowie *FingerprintContainer*. Ein Objekt vom Typ *Fingerprint* ist eine abstrakte Repräsentation einer Webseite und der *FingerprintContainer* bildet den Pool der bekannten Webseiten. In der Applikation darf nur ein einziges *FingerprintContainer*-Objekt existieren, dies wird durch die Verwendung des so genannten "Singleton Patterns" in dieser Klasse erreicht. Zwischen den beiden Klassen dieses Pakets besteht eine Kompositionsbeziehung (häufig auch strenge Aggregation genannt), ein *FingerprintContainer* kann beliebig viele

Fingerprint Objekte enthalten, während ein *Fingerprint*-Objekt nur in genau einem *FingerprintContainer*-Objekt enthalten sein kann.

Abbildung 5.3 zeigt die Inhalte des Pakets *Forms*. Auch hier bestehen Kompositionsbeziehungen zwischen den beiden Klassen *Form* und *FormContainer* sowie *FormElement* und *Form*. Dies bringt zum Ausdruck, dass in einem *FormContainer* zumindest ein Objekt vom Typ *Form* enthalten sein muss, das wiederum wenigstens ein *FormElement*-Objekt enthalten muss. Die beiden Klassen *CompareForms* und *CompareFormElements* werden beim Vergleich zweier Fingerabdrücke miteinander von der Klasse *FingerprintChecker* aufgerufen und haben die Aufgabe die in zwei Fingerabdrücken enthaltenen *Form*-Objekte bzw. die darin enthalten *FormElement*-Objekte miteinander zu vergleichen.

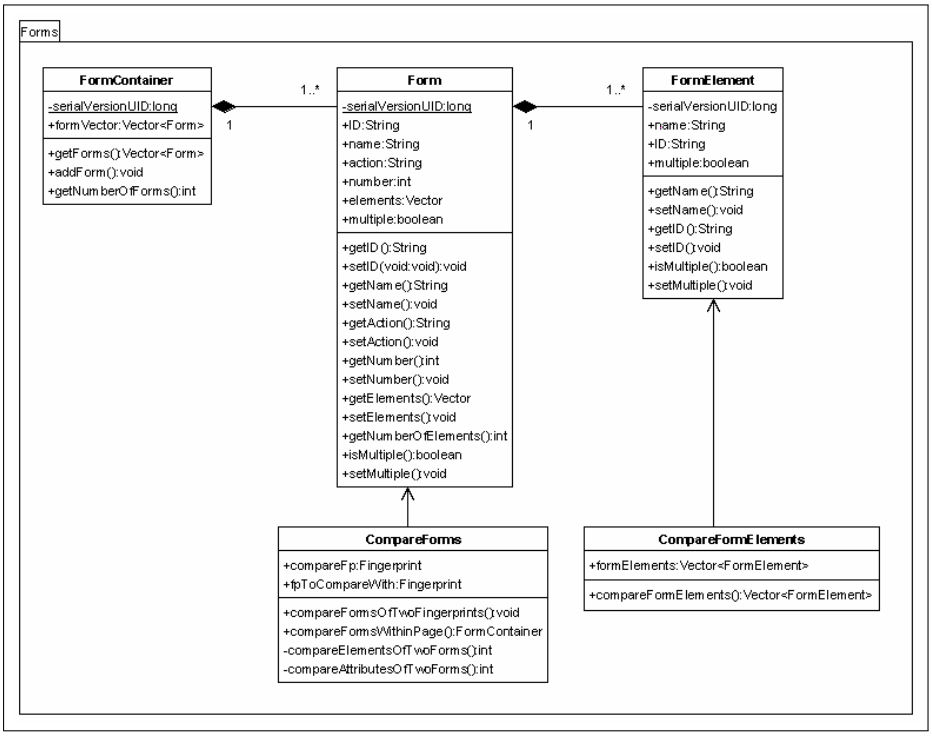


Abbildung 5.3: Forms-Paket

Das in Abbildung 5.4 dargestellte Paket *IDs* enthält die Klassen zur Verarbeitung der HTML-Tags mit ID-Attribut und wurde der Einfachheit halber zu "ID" abgekürzt. Auch hier besteht wieder eine Kompositionsbeziehung zwischen den Klassen *ID* und *IDContainer*. Ein Objekt vom Typ *ID* ist somit genau einem Objekt vom Typ *IDContainer* zugewiesen das wiederum nur dann existiert wenn es zumindest ein *ID*-Objekt enthält. Die Klasse *CompareIDs* wird ebenfalls aus der Klasse *FingerprintChecker* aufgerufen und vergleicht alle enthaltenen *ID*-Objekte zweier *IDContainer*-Objekte miteinander.

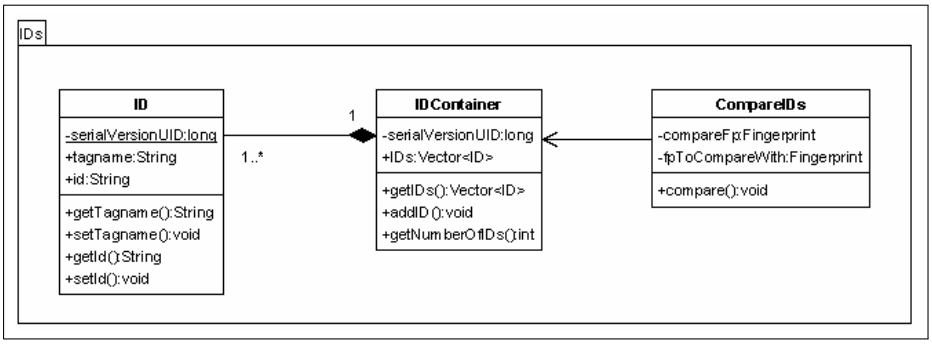


Abbildung 5.4: IDs-Paket

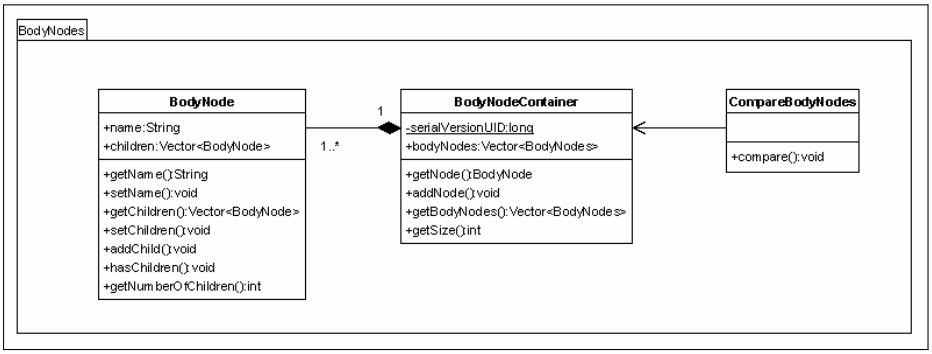


Abbildung 5.5: BodyNodes-Paket

Abschließend zeigt Abbildung 5.5 noch die Zusammensetzung des *BodyNodes* Paketes. Analog zum Paket *IDs* existieren auch hier eine Kompositionsbeziehung zwischen *BodyNodes* und *BodyNodeContainer*, sowie eine *CompareBodyNodes* Klasse die *BodyNode*-Objekte aus dem *BodyNodeContainer* ausliest um die enthaltenen HTML-Body-Teilbaum-Strukturen zweier Fingerabdrücke miteinander in Beziehung zu setzen. Auch in diesem Fall ist die aufrufende Klasse wieder *FingerprintChecker*.

5.3 Eingesetzte Fremdentwicklungen

Der einzige Bestandteil der Applikation der auf nicht öffentlich verfügbaren Fremdentwicklungen beruht stellt eine zwar nicht unwesentliche aber trotzdem umfangreiche Funktionalität zur Verfügung. An verschiedenen Stellen im Code wird auf eine Programmbibliothek zugegriffen, die von der Lixto Software GmbH zur Verfügung gestellt wurde. Die daraus verwendeten Funktionalitäten beschränken sich auf die Ermöglichung der Absetzung von XPath Abfragen auf den zu einem DOM-Dokument umgewandelten HTML-Code der aufgerufenen Webseite. Die XPath-Funktion hätte jedoch auch mit nicht proprietären Programmbibliotheken umgesetzt werden können, wie dem von Sun zur Verfügung gestellten XPath Interface [3].

[3]http://java.sun.com/j2se/1.5.0/docs/api/javax/xml/xpath/XPath.html

Kapitel 6

Evaluierung

Die Evaluierung der Anwendung soll zeigen dass die gewünschte Funktionalität auch tatsächlich erfolgreich umgesetzt wurde. Zu diesem Zweck wurden Testfälle entwickelt und angewendet, die zusammen mit den Testergebnissen in diesem Kapitel beschrieben werden. Darüber hinaus enthält es einen Abschnitt darüber wie der erstellte Programmcode verbessert, weiterentwickelt und optimiert werden könnte.

6.1 Testfälle

Um die Vergleichsfunktion des Systems zu testen wurden alle Webseitenkategorien der acht Quellseiten aus Tabelle 3.1 an das System übergeben und als Fingerabdrücke im Pool der bekannten Webseiten gespeichert.

Davor wurden zunächst alle Quellen mit den selben Parameter abgefragt:

Stadt: Wien
Ankunft: 01. Oktober 2009
Abreise: 05. Oktober 2009
Zimmeranzahl: 1
Erwachsene: 2
Kinder: 0

Nachdem von jeder Webseite die drei *Kategorien* Formular-, Übersichts- und Detailseite im Pool gespeichert wurden, ergeben sich aus den acht verwendeten Quellen 24 angelegte Fingerabdrücke. Bei den Übersichtsseiten wurde immer die jeweils erste angezeigte Seite für den Fingerabdruck verwendet und alle diese Test-Fingerabdrücke wurden am 01. Oktober 2009 angelegt.

Um feststellen zu können ob der Vergleich von Webseiten mit dieser Methode funktioniert wurden am 02. Oktober 2009 alle verwendeten Quellen erneut herangezogen und jede von ihnen noch einmal mit für alle Anbieter gleichen Parametern abgefragt:

Stadt: Bangkok
Ankunft: 14. November 2009
Abreise: 21. November 2009
Zimmeranzahl: 1
Erwachsene: 2
Kinder: 0

Von jeder Quellseite wurden alle drei Seitenkategorien mit der Aufgabe sie im Pool wiederzufinden an die Anwendung übergeben. Wie die Ergebnisse dieser Vergleiche aussehen kann in Tabelle 6.1 nachgelesen werden.

Von links nach rechts sind die Tabellen in folgenden Spalten aufgebaut:

1. Kategorie der als Input übergebenen und im Pool wieder zu findenden Webseite
2. Ergebnis des Vergleichs
   true - die korrekte Webseite wurde gefunden
   false - eine falsche Webseite wurde als Ergebnis geliefert
3. Die Gesamtpunktezahl des Fingerabdruck der - als dem Input entsprechend ausgegeben - wurde (Sieger genannt)
4. Punkte des Siegers auf URL Ebene
5. Punkte des Siegers auf Formular Ebene
6. Punkte des Siegers auf HTML-Tag mit ID Ebene
7. Punkte des Siegers auf HTML-Body-Teilbaum Ebene
8. Fingerabdruck aus dem Pool mit der zweithöchsten Gesamtpunktezahl, also der dem gesuchten als am zweitähnlichsten eingestufte Fingerabdruck
9. Gesamtpunkteanzahl des Fingerabdrucks mit den meisten Punkten nach dem "Sieger"

In den Zeilen befinden sich ganz links jeweils die Informationen zur Domäne einer Vergleichsgruppe sowie darunter die Kategorie der im Pool zu findenden Webseite.

Nach dem ersten Testdurchlauf wurden noch drei weitere gleichartige Tests durchgeführt, deren Ziel es war die Funktionalität des Konzepts bzw. der Anwendung an sich, sowie ihr Verhalten im Zeitverlauf zu beobachten. Die zwischen den einzelnen Tests liegenden Zeitspannen haben keine spezielle Bedeutung. Getestet wurde jeweils an Donnerstagen bzw. Sonntagen die der Entwicklung der Arbeit gewidmet waren.

Eine Woche nach Anlegen des Fingerabdruck-Pools wurden erneut alle vorhandenen Quellen abgefragt und deren URLs an die Applikation übergeben um im Pool den dem Input entsprechenden Fingerabdruck wieder zu finden. Die verwendeten Parameter dieser Tests finden sich in der folgenden Aufstellung, die dazugehörigen Ergebnisse in Tabelle 6.2.

Stadt: Amsterdam
Ankunft: 05. Dezember 2009
Abreise: 12. Dezember 2009
Zimmeranzahl: 1
Erwachsene: 2
Kinder: 0

Zehn Tage nach den vorangegangenen Tests wurde am 18. Oktober 2009 erneut ein Vergleich aller Quellen durchgeführt. Bei diesem Testfall wurden erneut andere Parameter als bei den vorherigen verwendet, diese waren:

Stadt: New York City
Ankunft: 23. Oktober 2009
Abreise: 25. Oktober 2009
Zimmeranzahl: 1
Erwachsene: 2
Kinder: 0

Im Zeitraum zwischen dem zweiten und dritten Testdurchgang wurde die Webseite www.hotelclub.com mit grundlegend verändertem Design neu veröffentlicht. Die offensichtlichste Änderung findet sich auf den Übersichtsseiten, die heute in einem sehr modernen Design mit Bildern für jedes angebotene Hotel aufwarten. Davor waren die selben Angebote in eine lange Liste eingebettet die weniger Details zu den Angeboten enthielt und auch weniger übersichtlich war. Dasselbe gilt für die Kategorie Detailseite und auch die Suchseite wurde grafisch erneuert. Im Gegensatz zu den alten Designs beispielsweise auch die Anzahl der enthaltenen Formulare geändert, so sind in der Suchseite heute nur noch zwei statt früher vier und in der Detailseite nur noch zwei statt früher sieben <form>-Konstrukte enthalten. In den Übersichtsseiten hat sich die Anzahl der Formulare zwar nicht geändert, die Inhalte dieser jedoch sehr wohl.

Bevor der dritte Testlauf durchgeführt wurde, wurden also die im Pool befindlichen Fingerabdrücke der drei Seitenkategorien von Hotelclub auf den neuesten Stand gebracht. Im selben Zeitraum wurde auch die Website von Ebookers Veränderungen unterzogen. Diese waren aber nicht so umfangreich und offensichtlich wie bei Hotelclub. Viel mehr wurde www.ebookers.com nur einer leichten Anpassung des Designs unterzogen ohne weitreichende strukturelle Änderungen im HTML-Code der einzelnen Seitenkategorien durchzuführen. Im Gegensatz zu Hotelclub wurden die bereits bestehenden Pool-Fingerabdrücke auch unverändert belassen. Trotzdem jedoch konnten die gesuchten Webseiten korrekt im Pool identifiziert werden. Das lässt den Schluss zu, dass die Überarbeitung der Webseite keine strukturell tiefgreifende war und eine Erneuerung der Pool-Fingerabdrücke immer dann notwendig wäre, wenn eine oder mehrere der in Fingerabdrücken verwendeten Webseitenkomponenten grundlegend verändert wurden. Leider steht keine Möglichkeit zur Verfügung die Unterschiede in der grafischen Darstellung der beiden Webseitenvarianten zur Illustration darzustellen. Tabelle 6.3 fasst diese Vergleichsergebnisse zusammen.

Im Falle von an die Anwendung übergebenen Übersichtsseiten wurde stets die letz-

te anzeigbare Seite mit Suchergebnissen gewählt. Dadurch soll möglichst eine andere Anzahl an angezeigten Hotels in der Webseite enthalten sein als in den für den Pool-Fingerabdruck verwendeten Webseiten der selben Kategorie. Die Quelle HRS präsentiert all ihre Ergebnisse in einer Liste die vollständig auf einer Webseite untergebracht ist, hier war dieses Vorgehen also nicht möglich.

6.2 Testergebnisse

Die Testergebnisse in den Tabellen 6.1 bis 6.3 zeigen, dass jeder Seite der korrekte Fingerabdruck aus dem Pool zugewiesen werden konnte. Die wenigsten Punkte als "Gewinner" eines Vergleichs hat die Expedia-Übersichtsseite mit 57; gleichzeitig hat diese Seite mit nur vier Punkten auch den geringsten Vorsprung zum zweitplatzierten Fingerabdruck. Diese geringe Punkteanzahl ist darauf zurückzuführen, dass die verglichenen URLs nur in 30 Buchstaben übereinstimmten und auch die sehr wenige übereinstimmende HTML-Tags mit ID-Attribut gefunden wurden. Für Formulare, bzw. den HTML-Body-Teilbaum konnten gar keine Übereinstimmungen gefunden werden was in null Punkten für diese beiden Komponenten resultierte. Dieses Ergebnis impliziert, dass sowohl die Vergleichslogik für Formulare als auch für den HTML-Body-Teilbaum noch Verbesserungsbedarf haben.

Der Fingerabdruck mit dem höchsten Punktestand findet sich ebenfalls in der Expedia-Gruppe; die Expedia-Formularseite hat mit 503 Punkten 178 Punkte mehr als die Webseite mit der zweithöchsten Punkteanzahl für einen korrekten Vergleich (HRS-Übersichtsseite). Darüber hinaus hat die Expedia-Formularseite mit einem Vorsprung von 429 Punkten auch gegenüber dem als am zweitähnlichsten eingestufen Fingerabdruck (Expedia-Detailseite) den größten Vorsprung aufzuweisen. Das heißt, dass die Vergleichslogik im Einsatz mit Expedia-Webseiten sowohl die überzeugendsten als auch die am wenigsten aussagekräftigen Ergebnisse liefert, jedoch ohne dabei eine falsche Zuordnung getroffen zu haben.

In den die Vergleichsergebnisse enthaltenden Tabellen zeigen sich einige weitere interessante Dinge. So kann man feststellen, dass die Formularseiten der einzelnen Anbieter den besten "Wiedererkennungswert" aufweisen. Ihre Gesamtpunkteanzahlen sind sehr stabil und schwanken bei den Vergleichen der Webseiten HRS, Travelocity und Venere gar nicht. Minimale Veränderungen weist die Expedia Formularseite auf und Booking zeigt erst in Tabelle 6.3 Unterschiede zu den vorherigen Vergleichen. Analog verhält es sich mit Hotels.com, wo die Ergebnisse der ersten beiden Tests konstant waren, der dritte Durchgang jedoch drei Punkte Unterschied auf Ebene der HTML-Tags mit ID aufweist.

Die beiden, während der Testläufe überarbeiteten Webseiten von Hotelclub bzw. Ebookers, lassen sich hier leider nicht besonders gut mit den anderen Quellen vergleichen. Ebookers fällt dabei jedoch auf, weil die Vergleiche trotz überarbeiteter Webseite korrekt durchgeführt werden konnten.

Die relative Konstanz der Ergebnisse bei der Suche nach Formularseiten lässt sich leicht erklären. Die Webseite auf der Benutzer ihre Suchparameter eingeben ist im allgemeinen weniger komplex als die anderer Kategorien. Ihre Inhalte sind als besonders gleichbleibend zu bezeichnen, da sie unabhängig von den eingegebenen Parametern sind und bei weitem weniger HTML-Code umfassen als beispielsweise eine Übersichtsseite.

Die anderen beiden untersuchten Seitenkategorien weisen in Relation zu den Formularseiten größere Schwankungen in den erreichten Gesamtpunktezahlen auf. Es gibt jedoch auch hier mit den Übersichtsseiten von HRS und Travelocity zwei Webseiten die in allen drei Tests die selbe Punkteanzahl erreichen konnten. Damit sind diese beiden Webseiten die einzigen zwei die absolut konstante Ergebnisreihen in zwei unterschiedlichen Webseitenkategorien aufzuweisen haben. Die größten Änderungen in der Punkteanzahl weißt die Hotelclub-Übersichtsseite auf, jedoch wurde die Webseite wie erwähnt zwischen dem zweiten und dritten Testlauf mit neuem Design veröffentlicht, deshalb bleibt diese Entwicklung unberücksichtigt. Von den Webseiten die keine grundlegende Überarbeitung während der Testläufe erfahren haben hat die Suche der HRS-Detailseite beim zweiten und dritten Testlauf gegenüber dem ersten am meisten Punkte eingebüßt, nämlich 101. Beim ersten Test wurde diese Webseite noch mit 173 Punkten wiedererkannt, während in den beiden folgenden Versuchen nur noch 72 Punkte erreicht werden konnten. Während der Wert für die Gleichheit der URLs konstant geblieben ist, wurde nur mehr eines statt zwei Formulare als gleich interpretiert (5 Punkte weniger), statt 83 nur mehr 12 HTML-Tags mit ihrem ID-Attribut wiedererkannt (-69 Punkte) und auch der HTML-Body-Teilbaum konnte nicht als gleich bezeichnet werden (-25 Punkte). Der Vergleich wurde jedoch trotz dieses großen Punkteverlustes erfolgreich durchgeführt, da die Seitenkategorie die als zweites gereiht wurde (beide Male die Übersichtsseite) mit 20 Punkten immer noch so wenige Gemeinsamkeiten mit der gesuchten Webseite aufweist dass eine eindeutige Identifizierung problemlos möglich ist.

Im Zeitverlauf lässt sich erkennen, dass die Gesamtpunktezahl der Vergleichsgewinner häufiger ab- als zunimmt. Bei einigen Webseiten gibt es ein gewisses auf und ab, jedoch gibt es keine Webseite bzw. Domänen-Kategorie, die von Vergleich eins bis drei immer jeweils steigende Gesamtpunkteanzahlen des Gewinners aufweisen kann. Daraus ließe sich schließen, dass je mehr Zeit zwischen Erstellung eines Referenzfingerabdrucks und Vergleich dessen mit einer aktuellen Webseitenversion vergeht, die Gemeinsamkeiten der beiden Webseiten immer geringer werden. Dadurch wird eine erfolgreiche Wiedererkennung immer schwieriger, und um zu vermeiden dass nach einer längeren Zeitperiode einzelne Kategorien falsch wiedergefunden und zugewiesen werden, wäre anzuraten die Referenzfingerabdrücke in regelmäßigen Abständen zu erneuern. Eine Erneuerung der hinterlegten Fingerabdrücke wäre natürlich auch bei offensichtlichen Veränderungen im Design oder der Ablauflogik der einzelnen Webseiten notwendig um die Funktionalität der Applikation aufrecht zu erhalten.

Tabelle 6.1: Testergebnisse 1

input	**correct match**	**Punkte**					**zweit-ähnlichste**	**Punkte gesamt**
		gesamt	URL	Forms	IDs	Baum		
Booking							**Booking**	
Suchseite	true	284	75	20	164	25	Detailseite	82
Übersichtsseite	true	135	37	20	78	0	Detailseite	103
Detailseite	true	182	29	35	119	0	Übersichtsseite	103
Ebookers							**Ebookers**	
Suchseite	true	119	75	5	14	25	Übersichtsseite	84
Übersichtsseite	true	154	110	5	14	25	Suchseite	85
Detailseite	true	96	54	5	12	25	Übersichtsseite	18
Expedia							**Expedia**	
Suchseite	true	503	75	5	398	25	Detailseite	74
Übersichtsseite	true	57	30	0	27	0	Suchseite	53
Detailseite	true	322	104	30	163	25	Suchseite	74
Hotelclub							**Hotelclub**	
Suchseite	true	212	75	20	92	25	Übersichtsseite	47
Übersichtsseite	true	166	68	20	53	25	Detailseite	68
Detailseite	true	140	58	15	42	25	Übersichtsseite	68
Hotels.com							**Hotels.com**	
Suchseite	true	152	75	15	37	25	Übersichtsseite	62
Übersichtsseite	true	206	107	5	69	25	Suchseite	62
Detailseite	true	122	71	5	46	0	Übersichtsseite	57
HRS							**HRS**	
Suchseite	true	178	75	15	63	25	Übersichtsseite	77
Übersichtsseite	true	325	44	45	211	25	Suchseite	77
Detailseite	true	173	55	10	83	25	Übersichtsseite	21
Travelocity							**Travelocity**	
Suchseite	true	234	75	20	114	25	Detailseite	32
Übersichtsseite	true	136	41	5	90	0	Detailseite	111
Detailseite	true	189	80	10	74	25	Übersichtsseite	81
Venere							**Venere**	
Suchseite	true	156	75	10	46	25	Übersichtsseite	35
Übersichtsseite	true	242	25	10	182	25	Suchseite	35
Detailseite	true	228	31	5	167	25	Übersichtsseite	34

Tabelle 6.2: Testergebnisse 2

input	**correct match**	**Punkte**					**zweit-ähnlichste**	**Punkte gesamt**
		gesamt	URL	Forms	IDs	Baum		
Booking							**Booking**	
Suchseite	true	284	75	20	164	25	Detailseite	82
Übersichtsseite	true	199	99	20	80	0	Detailseite	105
Detailseite	true	226	29	35	162	0	Übersichtsseite	107
Ebookers							**Ebookers**	
Suchseite	true	119	75	5	14	25	Übersichtsseite	84
Übersichtsseite	true	126	110	5	11	0	Suchseite	57
Detailseite	true	70	53	5	12	0	Übersichtsseite	43
Expedia							**Expedia**	
Suchseite	true	501	75	5	396	25	Detailseite	74
Übersichtsseite	true	57	30	0	27	0	Suchseite	53
Detailseite	true	276	130	25	148	25	Suchseite	74
Hotelclub							**Hotelclub**	
Suchseite	true	206	75	15	91	25	Übersichtsseite	47
Übersichtsseite	true	160	67	20	48	25	Detailseite	67
Detailseite	true	137	55	15	42	25	Übersichtsseite	68
Hotels.com							**Hotels.com**	
Suchseite	true	152	75	15	37	25	Übersichtsseite	62
Übersichtsseite	true	148	108	5	40	0	Suchseite	63
Detailseite	true	122	71	5	46	0	Übersichtsseite	57
HRS							**HRS**	
Suchseite	true	178	75	15	63	25	Übersichtsseite	77
Übersichtsseite	true	325	44	45	211	25	Suchseite	77
Detailseite	true	72	55	5	12	0	Übersichtsseite	20
Travelocity							**Travelocity**	
Suchseite	true	234	75	20	114	25	Detailseite	32
Übersichtsseite	true	136	41	5	90	0	Detailseite	111
Detailseite	true	192	80	10	77	25	Übersichtsseite	81
Venere							**Venere**	
Suchseite	true	156	75	10	46	25	Übersichtsseite	35
Übersichtsseite	true	246	25	10	186	25	Detailseite	36
Detailseite	true	237	31	5	176	25	Übersichtsseite	34

Tabelle 6.3: Testergebnisse 3

input	**correct match**	**Punkte**					**zweit-ähnlichste**	**Punkte gesamt**
		gesamt	URL	Forms	IDs	Baum		
Booking							**Booking**	
Suchseite	true	273	75	15	158	25	Detailseite	80
Übersichtsseite	true	188	99	15	74	0	Detailseite	93
Detailseite	true	201	29	30	142	0	Übersichtsseite	100
Ebookers							**Ebookers**	
Suchseite	true	91	75	5	11	0	Übersichtsseite	55
Übersichtsseite	true	116	110	0	6	0	Suchseite	46
Detailseite	true	67	53	5	9	0	Übersichtsseite	13
Expedia							**Expedia**	
Suchseite	true	501	75	5	396	25	Detailseite	74
Übersichtsseite	true	74	30	0	44	0	Suchseite	57
Detailseite	true	295	103	30	162	0	Suchseite	73
Hotelclub							**Hotelclub**	
Suchseite	true	203	24	20	134	25	Detailseite	101
Übersichtsseite	true	267	75	10	157	25	Detailseite	130
Detailseite	true	209	45	10	129	25	Übersichtsseite	130
Hotels.com							**Hotels.com**	
Suchseite	true	149	75	15	34	25	Übersichtsseite	62
Übersichtsseite	true	147	107	0	40	0	Detailseite	63
Detailseite	true	165	71	5	64	25	Übersichtsseite	57
HRS							**HRS**	
Suchseite	true	178	75	15	63	25	Übersichtsseite	77
Übersichtsseite	true	325	44	45	211	25	Suchseite	77
Detailseite	true	72	55	5	12	0	Übersichtsseite	20
Travelocity							**Travelocity**	
Suchseite	true	234	75	20	114	25	Detailseite	32
Übersichtsseite	true	136	41	5	90	0	Detailseite	111
Detailseite	true	124	47	5	47	25	Übersichtsseite	74
Venere							**Venere**	
Suchseite	true	156	75	10	46	25	Übersichtsseite	35
Übersichtsseite	true	242	25	10	182	25	Suchseite	35
Detailseite	true	226	31	5	165	25	Übersichtsseite	34

Neben den Tests die mit den zur Entwicklung verwendeten Webseiten durchgeführt wurden, wurden auch noch zwei weitere Szenarien der Datenextraktion aus dem Deep Web berücksichtigt. Es wurden zwei Webseiten die online Elektronikartikel anbieten, sowie eine Website die Bücher übers Internet verkauft herangezogen.

Die beiden Händler für Elektronikartikel sind die *notebooksbilliger.de AG* [1] aus Deutschland (aus Platzgründen in der folgenden Tabelle mit "nbb" abgekürzt) sowie die britische *Comet Group plc* [2]. Der erwähnte Buchhändler ist das österreichische Unternehmen *Morawa Buch und Medien GmbH & Co* [3].

Der größte Unterschied zwischen den Webseiten der Tabelle 3.1 und den Webseiten dieser Testreihe besteht darin, dass die Hotelzimmer über ein Formular abgefragt werden können, während man in den hier verwendeten Webseiten über Klicks durch Menüs zu den gewünschten Informationen navigiert. Trotzdem gibt es natürlich auch in diesen Fällen Webseiten die den bekannten Kategorien entsprechen. Die Kategorie Suchseite wird jedoch in Startseite umbenannt, da die Webseite kein Formular zur Datenbanksuche zur Verfügung stellt sondern nur Menüs zur Filterung der anzuzeigenden Daten auflistet. Die Kategorien Übersichts- und Detailseite werden nicht umbenannt, sie entsprechen von der Konzeption her auch weitestgehend den entsprechenden Webseiten aus der Gruppe der Hotelzimmer-Anbieter.

Auch in den Testfällen dieser Webseiten konnte jeder Vergleich korrekt durchgeführt werden, die genauen Ergebnisse sind in Tabelle 6.4 enthalten.

[1] http://www.notebooksbilliger.de
[2] http://www.comet.co.uk
[3] http://www.morawa-buch.at

Tabelle 6.4: Testergebnisse 4

input	**correct match**	**Punkte**					**zweit-ähnlichste**	**Punkte gesamt**
		gesamt	URL	Forms	IDs	Baum		
nbb							**nbb**	
Startseite	true	173	75	20	25	25	Übersichtsseite	83
Übersichtsseite	true	106	42	25	39	0	Detailseite	58
Detailseite	true	93	32	30	31	0	Übersichtsseite	63
Comet							**Comet**	
Startseite	true	81	22	5	54	0	Übersichtsseite	67
Übersichtsseite	true	228	75	15	113	25	Startseite	92
Detailseite	true	121	42	15	64	0	Übersichtsseite	93
Morawa							**Morawa**	
Startseite	true	106	75	5	1	25	Übersichtsseite	32
Übersichtsseite	true	36	31	5	0	0	Startseite	31
Detailseite	true	79	49	5	0	25	Startseite	31

Kapitel 7

Verbesserungspotential

Die in dieser Arbeit vorgestellte Anwendung ist prototypisch implementiert und in diversen Bereichen verbesserungsfähig. Diese Bereiche betreffen sowohl den Java-Code selbst, als auch die eingesetzten Techniken zu Analyse und Vergleich einzelner Komponenten eines Fingerabdrucks. Darüber hinaus würden sich noch einige weitere Bestandteile von Webseiten dazu anbieten Berücksichtigung zu finden

Der umgesetzte Programmcode ist von erfahrenen Programmierern bestimmt in vielen Dingen noch verbesserbar. Beispielsweise sollten in der Klasse *Main.java* die eingegebenen Schlüsselwörter mittels einer *switch-case*-Anweisung verarbeitet werden statt über *if*-Abfragen. Oder der Vergleich von Formularen-Attributen, der derzeit mittels 27 *if*-Abfragen realisiert ist, würde sich ebenfalls für eine Überarbeitung und Optimierung anbieten.

Einige weitere Webseiteninhalte würden sich anbieten in den Vergleich mit einzufließen, so ließe sich die Verwendung von Frames in einer Webseite im Fingerabdruck abbilden. Nicht uninteressant wäre beispielsweise auch ein Verfahren das Webseiten aufgrund der Positionierung von Inhalten miteinander vergleicht und feststellen kann ob ein bestimmter Inhalt wie eine Grafik oder ein Menü im selben Darstellungsbereich eines Browsers wiederzufinden ist oder nicht.

Die Analyse von Formularen könnten noch um Informationen zu ihrer Position im HTML-Baum erweitert werden. Dabei würde berücksichtigt werden wie die HTML-Struktur vom Wurzelknoten <html> bis zum jeweiligen Formular aufgebaut ist um in weiterer Folge auch in den Formularvergleich mit einzufließen. Außerdem könnte eine Miteinbeziehung von Informationen bzgl. der grafischen Positionierung eines Formulars im Browser von Vorteil sein einzelne Formulare eindeutig identifizieren und unterscheiden zu können.

Der Vergleich der einzelnen Komponenten miteinander ist ebenfalls noch verbesserungsfähig. Derzeit wird vieles sehr strikt gehandhabt, das heißt entweder etwas wird als völlig gleich angesehen oder als absolut ungleich, es gibt keine Zwischenstufen. So wird beim Vergleich des HTML-Teilbaumes absolute Gleichheit der beiden Bäume benötigt um Punkte für den beinhaltenden Fingerabdruck zu vergeben. Hier wäre ein

Mechanismus der auch teilweise gleiche Strukturen berücksichtigt und eine Maßzahl an Ähnlichkeit errechnen kann angebracht.

Darüber hinaus könnte die Bestimmung des den Vergleich gewinnenden Fingerabdrucks weiterentwickelt werden. Anstatt sich völlig auf die absolute Anzahl der erreichten Punkte zu verlassen wäre es angebracht eine relative Vergleichsgröße heranzuziehen. Denkbar wäre zum Beispiel die Verwendung eines "Goldstandards", der die Punkteanzahl enthält die vergeben würde wenn eine Webseite mit sich selbst verglichen wird. Man könnte die Vergleichsergebnisse mit der erreichten Zahl des Goldstandards in Beziehung setzen und so eine Punkteobergrenze definieren die nicht überschritten werden kann. Die Vergleichskandidaten würden nicht nach ihrer absoluten Punktezahl, sondern nach einer Größe gereiht werden die von den jeweiligen im Goldstandard eines Fingerabdrucks hinterlegten Punkten abhängig ist. Erhalten zwei Fingerabdrücke am Ende des Vergleichs in absoluten Zahlen gleich viele Punkte würden sie im momentanen Fall beide als Sieger des Vegleichs hervorgehen. Ein Miteinbeziehen einer oberen Punktegrenze würde es ermöglichen die erreichten Punkte eines Fingerabdrucks damit in Relation zu setzen und feststellen zu können welcher der beiden Fingerabdruck näher an seinen Goldstandard herangekommen und damit eher als Gewinner des Vergleichs zu betrachten ist. Vorausgesetzt dass sich die beiden Fingerabdrücke mit gleicher Punkteanzahl in ihren Goldstandards unterscheiden, könnte so eine bessere Zuordnung des Vergleichsgewinners getroffen werden.

Im Zusammenhang mit der Umsetzung eines Goldstandards stellt sich jedoch unter anderem die Frage wie in einem solchen Fall mit der URL umzugehen wäre. Die Übersichtsseiten von Expedia haben URLs mit jeweils mehr als 850 Zeichen und einer großen Zahl eingeschlossener Parameter. Das würde für den Goldstandard einer solchen Webseite bedeuten, dass dieser auf URL-Ebene bedeutend höher wäre als es die bei einem Vergleich zweier nicht absolut identischer Expedia-URLs erreichbaren Punkte jemals sein können. Dadurch wären im Vergleich zu URLs die keine offen sichtbaren Parameter enthalten und im Allgemeinen bedeutend kürzer sind die Ergebnisse verzerrt, da eine Expedia URL der korrekten Seitenkategorie bei 50 gleichen Zeichen von 850 möglichen nur eine Gleichheit von 5,88 % aufweisen würde. Wird aber die selbe URL mit einer URL einer völlig anderen Webseite mit Länge 30 verglichen, so wäre allein durch den Präfix "http://www." bereits eine grundsätzliche Gleichheit von 36,66 % gegeben.

Eine weitere Möglichkeit die Vergleiche zu optimieren und deren Zuverlässigkeit zu erhöhen wäre es nicht nur Pluspunkte für Ähnlichkeit, sondern auch Minuspunkte für Unterschiede zu vergeben. So wäre es vorstellbar beispielsweise im Bereich der HTML-Tags mit ID-Attribut nicht nur in beiden verglichenen Fingerabdrücken gefundene positiv, sondern auch nicht enthaltene negativ zu bewerten. Dies würde die Dominanz der *ID*-Ojekte - wie zum Beispiel bei den im vorigen Kapitel gezeigten Ergebnissen der Vergleiche von Expedia-Webseiten - reduzieren und vermutlich eine ausgewogenere Punkteverteilung ergeben.

Des weiteren wäre es von Vorteil eine Möglichkeit zu haben mehrere - prinzipiell als gleich zu betrachtende - Webseiten einem Fingerabdruck zuzuweisen. Aus den Unterschieden der einzelnen übergebenen Webseiten ließen sich variable Bestandteile fest-

stellen und auch als solche deklarieren. Dadurch würde sich sowohl die Abbildung der Webseite als Fingerabdruck präziser gestalten, als auch die Möglichkeiten zur Umsetzung des Vergleichs verbessern. Wird nämlich zur Hinterlegung im Pool eine Webseitenvariante gewählt die nicht alle möglichen Bestandteile enthält, so wird auch der Vergleich damit nie völlig richtig durchführbar sein.

Abbildungsverzeichnis

1.1 Beispielseite 1 7
1.2 Beispielseite 2 7
1.3 Beispielseite 3 7

3.1 Booking Stadtauswahl Ausschnitt 19
3.2 Suchseite 20
3.3 Ergebnisseite 20
3.4 Detailseite 20

5.1 Fingerprintchecker-Paket 52
5.2 Fingerprints-Paket 53
5.3 Forms-Paket 54
5.4 IDs-Paket 55
5.5 BodyNodes-Paket 55

Tabellenverzeichnis

3.1 Beispielquellen 18
3.2 Anzahl enthaltener Formulare 24
3.3 Anzahl enthaltener Tabellen 27
3.4 Durchschnittliche Anzahl enthaltener Tags mit ID-Attributen 28

4.1 Punktevergabe 41

6.1 Testergebnisse 1 62
6.2 Testergebnisse 2 63
6.3 Testergebnisse 3 64
6.4 Testergebnisse 4 66

Listings

4.1 compareFormsWithinPage 33
4.2 compareElementsOfTowForms 34
4.3 getIDs 37
4.4 getBodyNodes 39
4.5 compareURLs 43
4.6 compareIDs 45
4.7 compareBodyNodes 47

Literaturverzeichnis

[1] R. Baumgartner, O. Frölich, G. Gottlob, "The Lixto Systems Applications in Business Intelligence and Semantic Web," *Lecture Notes in Computer Science, The Semantic Web: Research and Applications*, pp. 16–26, 2007.

[2] Wikipedia, the free encyclopedia, "Web crawler."

[3] S. Flesca, G. Manco, E. Masciari, E. Rende, A. Tagarelli, "Web wrapper induction: a brief survey," *AI Communications Vol. 17/2*, 2004.

[4] C. Chang, M. Kayed, M. Girgis, K. Shaalan, "A Survey of Web Information Extraction Systems," *IEEE Transactions on Knowledge and Data Engineering, TKDE-0475-1104.R3*, 2006.

[5] A. H. F. Laender, B. A. Ribeiro-Neto, "A Brief Survey of Web Data Extraction Tools," *SIGMOD Record, Vol. 31*, pp. 84 – 93, 2002.

[6] R. Baumgartner, M. Ceresna, G. Ledermüller, "Deep Web Navigation in Web Data Extraction," *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce Vol-2 (CIMCA-IAWTIC'05)*, pp. 698 – 703, 2005.

[7] V. Bush, "As we may think," *The Atlantic Monthly*, 1945.

[8] SEW Staff, "Nielsen NetRatings Search Engine Ratings - Search Engine Watch (SEW)."

[9] C. N. Mooers, "Making information retrieval pay," *118th Meeting, Abstracts of Papers, American Chemical Society.*

[10] G. Salton and M.J. McGill, *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.

[11] M. K. Bergmann, "The Deep Web: Surfacing Hidden Value." The journal of electronic publishing, Vol. 7, No. 1, 2001.

[12] K. C. C. Chang, B. He, C. Li, M. Patel, Z. Zhang, "Structured Databases on the Web: Observations and Implications," *ACM SIGMOD Record, Vol. 3, Issue 3*, pp. 61 – 70, 2004.

[13] L. Eikvil, "Information Extraction from World Wide Web - A Survey." Technical Report 945, Norwegian Computing Center, 1999.

[14] D. Shen, Z. Chen, Q. Yang, H. Zeng, B. Zhang, Y. Lu and W. Ma, "Web-page Classification through Summarization," *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 242–249, 2004.

[15] A. L. Berger and V. O. Mittal, "OCELOT: A system for summarizing web pages," *Procedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 144–151, 2000.

[16] Wikipedia, the free encyclopedia, "Natural language processing."

[17] V. Anupam, J. Freire, B. Kumar, D. Lieuwen, "Automating Web navigation with the WebVCR," *Computer Networks 33*, 2000.

[18] A. Le Hors, P. Le Hegaret, L. Wood, G. Nicol, J. Robie, M. Champion, S. Byrne, "Document Object Model (DOM) Level 3 Core Specification." http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/, 2007.

[19] J. Carme, M. Ceresna, O. Frölich, G. Gottlob, T. Hassan, M. Herzog, W. Holzinger, B. Krüpl, "The Lixto Project: Exploring New Frontiers of Web Data Extraction," *Proceedings of the 23rd British National Conference on Databases*, 2006.

[20] G. Gottlob, C. Koch, R. Baumgartner, M. Herzog, S. Flesca, "The Lixto Data Extraction Project - Back and Forth between Theory and Practice," *Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems.*

[21] A. Berglund, S. Boag, D. Chamberlin, M F. Fernandez, M. Kay, J. Robie, J. Simeon, "XML Path Language(XPath) 2.0."

[22] M. Kowalkiewic, M. Orlowska, T. Kaczmarek, W. Abramowicz, "Robust Web Content Extraction," *Proceedings of the 15th international conference on World Wide Web, WWW*, pp. 887–888, 2006.

[23] J. Myllimaki, J. Jackson, "Robust Web Data Extraction with XML Path Expressions," 2002.

[24] T. Lukasser, "Efficient Processing of XPath Queries." Masters Thesis, Institut für Informationssysteme, TU Wien, 2004.

[25] A. Watt, *Beginning Regular Expressions (Programmer to Programmer).* Wiley Publishing Inc., 2005.

[26] Wikipedia, the free encyclopedia, "Formale Sprache."

[27] Wikipedia, the free encyclopedia, "Formale Grammatik."

[28] Wikipedia, the free encyclopedia, "Produktionsregel."

[29] Wikipedia, the free encyclopedia, "Webbrowser."

[30] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms.* MIT Press, 1990.

[31] D. Ragett, A. Le Hors and I Jacobs, "Html 4.01 specification."

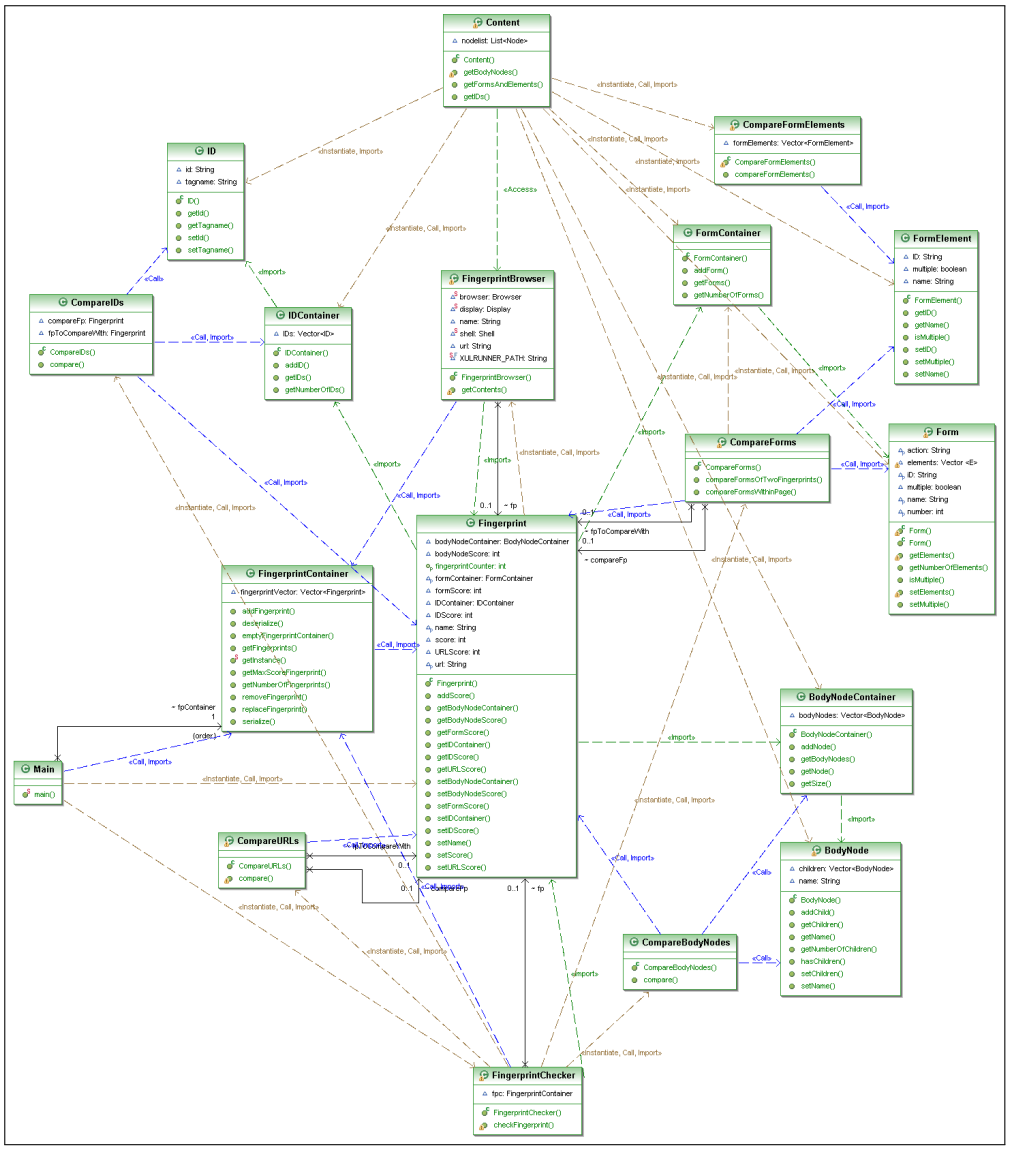
[32] Wikipedia, the free encyclopedia, "Baum (Graphentheorie)."

[33] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau, J. Cowan, "Extensible Markup Language (XML) 1.1 (Second Edition)."

[34] Wikipedia, the free encyclopedia, "Erweiterte Backus-Naur-Form."

Anhang A

Klassendiagramm



Anhang B

Installationsanleitung

Entpacken Sie das Archiv Applikation.zip und speichern Sie die gesamten Inhalte an einem beliebigen Ort.

Beispiel - C:\Applikation

Öffnen Sie ein Command-Fenster (Start-Ausführen-"cmd"-OK)

Wechseln Sie in den Ordner in den Sie die Applikation entpackt haben.

Beispiel - cd C:\Applikation

Führen Sie das script install.bat aus.

Beispiel - C:\Applikation>install.bat

Die Applikation kann nun benutzt werden, folgende Eingabeoptionen stehen zur Verfügung:

"java -jar Fingerprintchecker.jar info"
Zeigt diese Benutzungsregeln an.

"java -jar Fingerprintchecker.jar list"
Listet alle im Pool enthaltenen Fingerabdrücke auf.

"java -jar Fingerprintchecker.jar add [Name] [URL]"
Legt für die übergebene URL einen Fingerabdruck an und fügt ihn mit dem übergebenen Namen zum Pool hinzu.

"java -jar Fingerprintchecker.jar replace [Name] [URL]"
Überschreibt den im Pool unter "Name" hinterlegten Fingerabdruck.

"java -jar Fingerprintchecker.jar remove [Name]"
Löscht den Fingerabdruck mit dem übergebenen Namen aus dem Pool.

”java -jar Fingerprintchecker.jar clear”
Löscht alle im Pool hinterlegten Fingerabdrücke.

”java -jar Fingerprintchecker.jar find [URL]”
Vergleicht die Webseite der übergebenen URL mit allen im Pool hinterlegten Fingerabdrücken und bestimmt den der Eingabe am ähnlichsten

Anhang C

Kurzfassung

In den vergangenen Jahren wurden viele Versuche unternommen im Internet verfügbare Webseiten automatisch in verschiedenste Kategorien einzuteilen. Dabei war das bisherige Vorgehen dadurch geprägt diese Webseiten anhand ihrer Inhalte in Klassen von Angeboten zu kategorisieren. Solche Kategorien bzw. Klassen sind beispielsweise *Universitäts-Seiten*, die *Internet-Auftritte von Unternehmen*, *Webshops*, *Web-Suchmaschinen*, *Adressverzeichnisse* usw. Dazu bediente man sich verschiedenster Vorgehensweisen, wie der Analyse der Hyperlink-Struktur innerhalb der Website und den ein- und ausgehenden Verbindungen, oder den in den Webseiten enthaltenen Informationen der Meta-Tags. Diese Einteilungen sollten vornehmlich Anbietern von Suchdiensten im World Wide Web (WWW) dienen um kategorisierte Ergebnisse für Suchanfragen liefern zu können. Darüber hinaus wurden auch einige Anstrengungen unternommen auf automatische Weise textbasierte Zusammenfassungen von Webseiten zu erstellen um die Seiten dadurch kompakt und möglichst aussagekräftig zu beschreiben.

Die vorliegende Arbeit beschäftigt sich ebenfalls mit Bestandteilen von Webseiten, jedoch nicht mit dem Ziel Webseiten zusammenfassen oder zu versuchen ihre Inhalte zu verstehen, sondern damit so genannte *Fingerabdrücke* von ihnen zu erstellen. Ein Fingerabdruck ist eine abstrakte Repräsentation einer Webseite, basierend auf ausgewählten Bestandteilen des ihr zu Grunde liegenden *Hypertext Markup Language (HTML)* Codes. Mit Hilfe des Fingerabdrucks einer bestimmten Webseite soll es einem Computer ermöglicht werden diese Webseite automatisiert wiederzuerkennen, sowie mit dieser Webseite inhaltlich vergleichbare Seiten als solche zu identifizieren indem die Fingerabdrücke der betreffenden Webseiten miteinander verglichen werden. Als Anwendungsszenario für diese Technologie dient die automatische Navigation durch Webseiten zur Extraktion von Inhalten. Es gibt eine große Anzahl von so genannten *Webcrawlern*, häufig auch als *Wrapper* bezeichnet, deren Aufgabe darin besteht automatisch durch verschiedene Webseiten zu navigieren um dort bestimmte Inhalte strukturiert zu extrahieren. Die vorgestellte Anwendung versteht sich als Erweiterung eines Webcrawlers, die dazu in der Lage ist vom Wrapper zu bearbeitende Webseiten mittels ihrer Repräsentation als Fingerabdruck zu speichern. Ein Wrapper hat im Allgemeinen mehrere unterschiedliche Webseiten in einer ganz bestimmten Reihenfolge zu bearbei-

ten für die jeweils genau definierte Regeln zu Navigations- oder Extraktionsschritten festgelegt sind. Entspricht die Reihenfolge der aufgerufenen Webseiten während der Abarbeitung des Webcrawlers nicht der vorgesehenen Sequenz können die an diesem Punkt der Abfolge vorgesehenen Mechanismen zur Navigation oder Extraktion nicht mehr angewendet werden. Um eine Fortführung des Webcrawlers jedoch trotzdem zu ermöglichen wird von der zu diesem Zeitpunkt offenbar "falschen" Webseite ein Fingerabdruck erstellt und dieser mit den Fingerabdrücken der als bekannt gespeicherten Webseiten verglichen. Dieser "Pool der bekannten Webseiten" wurde vom Entwickler während der Erstellung des Wrappers angelegt und enthält Beispiele der vom Wrapper zu bearbeitenden Webseiten. Mit Hilfe eines Punktesystems wird festgestellt mit welcher Webseite der im sog. "Pool" gespeicherten die verglichene Webseite die größten Gemeinsamkeiten aufweist um jene Extraktions- oder Navigationsmechanismen aktivieren zu können, die für diese Art von Webseite vorgesehen sind.

Die Erstellung und der Vergleich von Fingerabdrücken wurde in einer Java-Applikation umgesetzt und verschiedene Tests haben gezeigt, dass die gewünschte Funktionalität gegeben ist, denn jeder durchgeführte Vergleich hat ein korrektes Ergebnis geliefert. Trotzdem jedoch hat die Applikation noch großes Verbesserungs- und Erweiterungspotential, dies betrifft vor allem die Ausweitung der verwendeten HTML-Komponenten sowie die Optimierung des verwendeten Punktesystems.

Als Basis für die durchgeführten Analysen und Tests dienten acht Websites die es ermöglichen online Hotelzimmer zu buchen, siehe Tabelle 3.1.

Die erstellte Applikation ist der Arbeit beigefügt.

Anhang D

Abstract

Over the past few years, numerous attempts to categorize webpages automatically have been described. Such categories are for example *University-Websites*, *Company-Websites*, *Onlineshops*, *Web-Search engines* and so on. To achieve this, many different approaches where proposed, such as analysing the hyperlink structure within a website and the connections to and from a webpage, or the information contained in the Meta-Tags of a webpage. This kind of categorization should mainly help search-engine operators of the World Wide Web (WWW) to be able to present categorized results of their searches. Besides that there were also quite some efforts to find ways to automatically summarize webpages to be able to describe these webpages as compact and meaningful as possible.

The present work also deals with contents of websites, but not with the goal to summarize or understand its contents, but to produce a so called *fingerprint* of them. A fingerprint is an abstract representation of a webpage based on selected components of the underlying *Hypertext Markup Language (HTML)* code. With the fingerprint of a certain webpage a computer shall be enabled to recognize this webpage in an automated manner, as well as webpages which have comparable contents and are to be viewed as alike, just by comparing the fingerprints of the webpages in question. The application scenario of this approach is the automated navigation through webpages to extract data. There is a great number of *webcrawlers*, often called *wrappers*, whose task is to automatically navigate through different webpages to extract some information of interest. The proposed application is to be viewed as an extension to a webcrawler that gives it the ability to save representations of webpages the wrapper has to deal with, as fingerprints. Generally a wrapper has to handle several different webpages in a very strict order and has well defined rules of navigation or extraction for each one of these webpages. If the sequence of called webpages during execution-time does not correspond to the scheduled order, the intended mechanisms for navigation or extraction cannot be executed at this point. To be able to keep up execution of a wrapper a fingerprint of the at this point ”wrong” webpage is made to compare it with the fingerprints of those webpages which have been saved as ”known webpages” before. This ”pool of known webpages"has been created by the developer during developing the wrapper and

contains examples of webpages a wrapper has to deal with. By using a scoring system it is being determined which one of the webpages contained in this "pool" has the most similarities with the webpage being compared, to be able to activate those navigation and extraction mechanisms which are intended for this kind of webpage.

Creation and comparison of fingerprints were implemented as a java application and several tests have shown that the desired functionality has been achieved, as every webpage comparison had correct results. Still the application can be improved in many ways, mainly by considering more HTML-components of webpages and by optimizing the scoring system.

All analysis and tests were based on eight websites which enable their users to book hotelrooms online, see table 3.1.

The developed application is attached to this work.

Anhang E

Lebenslauf

Persönliche Daten

Vorname	Wolfgang
Nachname	Jandl
Akademischer Grad	Bakk.rer.soc.oec
Anschrift	Stättermayergasse 32/10 1150 Wien
Telefon	0650/680 95 89
Email	A0209898@unet.univie.ac.at
Geburtsdatum	23. Juli 1981
Geburtsort	Klagenfurt
Familienstand	ledig

Ausbildung

1987 - 1991	Volksschule Bad Kleinkirchheim
1991 - 1996	Bundesrealgymnasium Spittal/Drau
1996 - 2001	Bundeshandelsakademie Spittal/Drau
2002 - 2008	Bakkalaureatsstudium Wirtschaftsinformatik Universität Wien und Technische Universität Wien
2008 - 2009	Masterstudium Wirtschaftsinformatik Universität Wien