Generierung von Sprachbefehlen mittels Datenextraktion aus Internetseiten

Robert Michael Maier

MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im Juni 2012

Erklärung

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

Hagenberg, am 25. Juni 2012

Robert Michael Maier

Inhaltsverzeichnis

Erklärung **iii**

Vorwort **viii**

Kurzfassung **ix**

Abstract **x**

1 Einleitung **1**
1.1 Zielsetzung 1
1.2 Forschungsfrage 1
1.3 Grundlegender Ansatz 1
1.4 Themen 2

2 Grundlagen und Stand der Technik **3**
2.1 Spracherkennung 3
2.1.1 Anwendungsbereiche 3
2.1.2 Sprachsignal 4
2.1.3 Methoden 5
2.1.4 Sprachmodelle 6
2.1.5 Grammatik 6
2.1.6 Google-Spracherkennung 9
2.2 Sprachein- und Sprachausgabe im Web 10
2.2.1 Sprachausgabe 10
2.2.2 Spracheingabe 10
2.3 Barrierefreiheit im Internet 11
2.3.1 Warum sie wichtig ist 11
2.3.2 Web Content Accessibility Guidelines 1.0 12
2.3.3 Web Content Accessibility Guidelines 2.0 15
2.3.4 WAI-ARIA 15
2.4 Volltextsuche 18
2.4.1 Apache Lucene 18
2.4.2 Zend Search Lucene 22
2.5 HTML-Spezifikation 22

2.5.1 Element-Gruppen 22
2.5.2 HTML5 24

3 Related Work **26**
3.1 The WAMI Toolkit 26
3.1.1 Komponenten und Funktionsweise 26
3.1.2 Abgrenzung 27
3.2 Online Speech Recognition API 27
3.2.1 Komponenten und Funktionsweise 27
3.2.2 Abgrenzung 28

4 Generierung der Sprachbefehle **29**
4.1 Ansatz 29
4.2 Auslesen der Textwerte 30
4.2.1 Links 30
4.2.2 Buttons 33
4.2.3 Formulare 33
4.2.4 Benutzerdefinierte Widgets 34
4.3 Nachbearbeitung der Werte 34
4.3.1 Ersetzen von Sonderzeichen 35
4.3.2 Entfernen von Anführungszeichen 35
4.3.3 Zahlen 36
4.4 Schlüsselwörter 37
4.4.1 Standard-Schlüsselwörter 37
4.4.2 Einflüsse auf die Schlüsselwörter 38
4.5 Formularelemente 39
4.5.1 Textfelder 39
4.5.2 Checkboxen 40
4.5.3 Select-Boxen und Radiobuttons 40
4.6 Zusätzliche Befehle 41
4.6.1 Vor- und Zurück-Navigation 42
4.6.2 Neu-Laden der Seite 43
4.6.3 Schließen des Fensters 43
4.6.4 Aufrufen der „Home“-Seite 44
4.6.5 Ausdrucken der Seite 44
4.7 Klassifizierung einer Internetseite 44
4.8 Beispiel BBC 45
4.8.1 Navigation 45
4.8.2 Top News Stories 46
4.8.3 Links zu Artikeln und Kategorien 48
4.8.4 „Most Popular in News“ 49
4.8.5 Suche 49
4.9 Beispiel Stackoverflow 52
4.9.1 Navigation 53

4.9.2 Links zu Fragen und Kategorien 53

5 Masterprojekt **54**
5.1 Projektbeschreibung 54
5.2 Warum Plug-In frei? 54
5.3 Technische Voraussetzungen 55
5.3.1 Clientseitig 55
5.3.2 Serverseitig 55
5.4 Systemarchitektur 55
5.4.1 Aufrufen einer Internetseite 56
5.4.2 Eingabe eines Sprachbefehls 57
5.4.3 Grammatikgenerator 57
5.4.4 Sprach-Controller 58
5.5 Interaktion mit einer Internetseite 59
5.5.1 Maus 59
5.5.2 Tastatur 60
5.6 Überwachen von Veränderungen 60
5.6.1 Bestehende Elemente 60
5.6.2 Erstellte oder Entfernte Elemente 61
5.7 Volltextsuche 63
5.7.1 MySQL 64
5.7.2 Apache Lucene 65
5.7.3 Apache Solr 66
5.7.4 Zend Search Lucene 66

6 Praktische Versuche **68**
6.1 Versuchsaufbau 68
6.2 Verwendete Internetseiten 68
6.2.1 BBC.com 68
6.2.2 Stackoverflow.com 69
6.3 Versuchsablauf 69
6.4 Seitenübergreifende Erkenntnisse 70
6.4.1 Verwendete Sprachbefehle 70
6.4.2 Scrollen 71
6.4.3 Suche 72
6.4.4 Fenster und Tabs 73
6.4.5 Markieren von Text 73
6.4.6 Aktivieren eines Seiten-Bereiches 74
6.4.7 Google-Spracherkennung 74
6.5 Erkenntnisse auf BBC.com 74
6.6 Erkenntnisse auf Stackoverflow.com 75
6.6.1 Verwendete Link-Texte 75
6.6.2 Erkennung von Abkürzungen 75
6.6.3 Verwendete Schlüsselwörter 75

6.7 Zusammenfassung 75

7 Schlussfolgerungen, Fragen und Ausblick 77
7.1 Allgemeine Schlussfolgerungen 77
7.2 Offene Fragen 78
7.2.1 Klassifikation der Internetseiten 78
7.2.2 Mikroformate 78
7.3 Mögliche Weiterentwicklungen 78
7.3.1 Performance 79
7.3.2 Klassifizierung 79
7.3.3 Navigation über mehrere Seiten 79

8 Zusammenfassung 80

Abbildungsverzeichnis 81

Tabellenverzeichnis 82

A Installationsanleitung des Masterprojektes 83
A.1 Voraussetzungen 83
A.2 Apache 2 HTTPD Server 83
A.2.1 Installation mit Tasksel 83
A.2.2 Manuelle Installation 84
A.2.3 Rewrite-Modul konfigurieren 84
A.3 Benötigte Skripte und Dateien installieren 84
A.3.1 Lucene 85

Quellenverzeichnis 86
Literatur 86
Online-Quellen 87

Vorwort

Diese Masterarbeit beschreibt die theoretischen Grundlagen, auf denen während der letzten beiden Semester meines Masterstudiums ein Projekt zur Sprachsteuerung von Internetseiten umgesetzt wurde. Während die Grundfunktionalität dieses Projektes bereits im ersten Semester funktionstüchtig war, flossen im Laufe der weiteren theoretischen Bearbeitung der Thematik noch weitere Erkenntnisse in das Projekt ein. Großes Augenmerk bei der Umsetzung des Projektes und der Masterarbeit war die Barrierefreiheit im Internet, die immer mehr in den Fokus der Web-Entwicklung gerät.

Bedanken möchte ich mich an dieser Stelle bei meinem Betreuer Mag. Volker Christian, der mich sowohl während der Projektumsetzung, als auch während dem Verfassen der Masterarbeit bestens unterstützte. Besonderer Dank gilt meinen Eltern, Johann und Elisabeth Maier, meinen Geschwistern, Alexandra und Johannes, meiner Freundin Martina und nicht zuletzt all meinen Freunden, die mir alle während meines Studiums unterstützend und verständnisvoll zur Seite standen.

Kurzfassung

Diese Masterarbeit beschreibt eine Möglichkeit, Internetseiten mittels Sprachbefehlen zu steuern. Dabei werden die Sprachbefehle automatisch aus der Struktur der Internetseite generiert. Dazu werden alle zur Interaktion mit einer Internetseite relevanten Elemente ausgelesen, um daraus einen oder mehrere Sprachbefehle zu generieren.

Um aus den Elementen die nötigen Informationen auslesen zu können, wird die HTML-Spezifikation untersucht und dabei Möglichkeiten aufgezeigt, schriftliche Repräsentationen und den Kontext eines Elementes zu extrahieren.

Zusätzlich wird ein System vorgestellt, das einige der erarbeiteten Theorien unterstützt und die Google-Spracherkennung zur Eingabe der Sprachbefehle verwendet. Anhand dieses Systems wurden einige Versuche durchgeführt und ausgewertet.

Abstract

This thesis describes a method to automatically generate voice commands to be able to use websites with natural language. These commands are generated based on the structure of the website for all HTML elements that are relevant to interact with a website.

In order to generate the commands, a textual representation is calculated based on each element's type and it's context within the website. This text value contains the name or description of an element and additional keywords to simulate the use of natural language.

Additionally, a system implementing some of the theories contained in this thesis is described. The implementation of the system architecture is used for real world tests. The results of these tests are then described and evaluated.

Kapitel 1

Einleitung

1.1 Zielsetzung

Diese Arbeit soll Aufschluss darüber geben, ob Sprachbefehle zur Steuerung von Internetseiten rein auf Basis von deren Struktur und Inhalt generiert werden können.

Die aufgestellten Theorien sollen anschließend durch praktische Versuche eines Prototypen überprüft werden. Die daraus gewonnen Erkenntisse sollen dann als Basis zur Erarbeitung von Weiterentwicklungsmöglichkeiten dienen.

1.2 Forschungsfrage

Ziel dieser Arbeit ist die Beantwortung der folgenden Forschungsfrage:

> Gibt es Möglichkeiten aus der Struktur einer Internetseite automatisch Sprachbefehle zur Sprachsteuerung der Seite zu extrahieren?

Dazu werden die folgenden Fragen in der vorliegenden Arbeit beantwortet?

- Welche Regeln zur Strukturierung von Internetseiten existieren?
- Durch welche Frameworks und Spezifikationen kann die Generierung der Sprachbefehle unterstützt werden?
- Welchen Einfluss hat die Art der Internetseite auf die Sprachbefehle?
- Können die generierten Befehle auch praktisch verwendet werden?

1.3 Grundlegender Ansatz

Der grundlegende Ansatz dieser Arbeit ist es, aus jedem HTML-Element, das zur Interaktion mit einer Internetseite verwendet wird, eine schriftliche Repräsentation zu berechnen. Dieser Text wird anschließend mit weiteren Wörtern ergänzt und mittels einer Volltextsuche indiziert. In die Auswahl

dieser Schlüsselwörter fließen verschiedene Faktoren ein, die im theoretischen Teil dieser Arbeit erläutert werden.

1.4 Themen

Die technischen Grundlagen zum Verständnis der Arbeit und der erarbeiteten Konzepte werden in Kapitel 2 beschrieben. Ein Überblick über die Spracherkennung wird in Abschnitt 2.1 gegeben. Dabei wird neben den Anwendungsbereichen und Methoden der Spracherkennung auch die formale Definition von Grammatiken beschrieben. In Abschnitt 2.2 werden die Möglichkeiten aufgezeigt, innerhalb eines Web-Browsers Audiosignale zu verarbeiten. Anschließend werden die Themen Barrierefreiheit (Abschnitt 2.3) und Volltextsuche (Abschnitt 2.4) erarbeitet, um abschließend in Abschnitt 2.5 die relevanten Teile der HTML-Spezifikation zusammenzufassen.

In Kapitel 3 werden zwei ähnliche Ansätze zur Sprachsteuerung von Internetseiten angeführt. Dabei wird auch die Abgrenzung zum Ansatz der vorliegenden Arbeit erörtert.

Weiters werden in Kapitel 5 Informationen über die Architektur und Funktionsweise des im Rahmen des Masterprojektes umgesetzten Systems zur Sprachsteuerung gegeben.

Kapitel 4 zeigt, wie Sprachbefehle aus Internetseiten generiert werden können und stellt den theoretischen Kernteil der Arbeit dar. Die erarbeiteten Generierungsregeln werden abschließend anhand von Beispielen erläutert.

Eine Versuchsreihe mit einigen Testpersonen wird in Kapitel 6 dokumentiert. Dabei wird der verwendet Versuchsaufbau, sowie die verwendeten Internetseiten beschrieben und einige weitere Informationen über den Ablauf der Tests aufgelistet.

Abschließend werden in Kapitel 7 die Schlussfolgerungen, offenen Fragen und Möglichkeiten zur Weiterentwicklung erarbeitet, Kapitel 8 fasst die Arbeit zusammen.

Kapitel 2

Grundlagen und Stand der Technik

In diesem Kapitel werden die theoretischen Hintergründe zum Verständnis dieser Arbeit sowie der aktuelle technische Stand der Forschung in den betreffenden Bereichen beschrieben. Zuerst wird eine allgemeine Einführung in die Spracherkennung gegeben. Anschließend werden die Möglichkeiten der Sprachverarbeitung im Internet erläutert, um abschließend die Barrierefreiheit im Allgemeinen sowie bezogen auf das Internet zu erläutern.

2.1 Spracherkennung

Die Spracherkennung als Teil der Sprachverarbeitung beschäftigt sich mit der Umwandlung von gesprochenem Text in eine entsprechende schriftliche Repräsentation und wird im englischen daher als „speech-to-text transcription“ bezeichnet [1, S. 299].

2.1.1 Anwendungsbereiche

Systemen, mit denen Menschen in natürlicher Sprache interagieren können, *„sind im Prinzip fast keine Grenzen gesetzt“* [10, S. 289]. Praktisch ist die Verwendung jedoch eingeschränkt, da die aktuell zur Verfügung stehenden Spracherkennungssysteme noch nicht ausgereift sind. Mit den aktuell am Markt befindlichen Systemen können jedoch bereits einige Aufgabengebiete abgedeckt werden [10, S. 289]:

1. *„Steuerung des Telefons, der Stereoanlage oder der Klimaanlage im Auto“*
2. *„Bedienungshilfen für Behinderte“*
3. *„Bedienung von Geräten im Operationssaal oder im Dunkelraum“*
4. *„Eingabe von medizinischen Befunden (Diagnose, Radiologie)“*

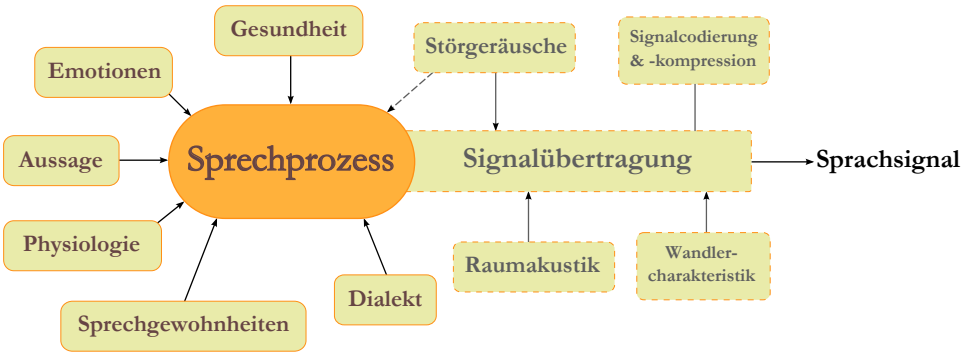


Abbildung 2.1: Einflüsse auf ein Sprachsignal [10, S. 25]

5. *„Bedienung von Miniaturgeräten (Handys, PDAs, Fernbedienungen)“*

Basierend auf dieser Auflistung kann die vorliegende Arbeit zum zweiten Anwendungsfall gezählt werden, indem körperlich beeinträchtigten Menschen die Bedienung einer Internetseite mit natürlicher Sprache ermöglicht wird.

2.1.2 Sprachsignal

Unter einem Sprachsignal versteht man die elektrische Repräsentation der Aussage eines Sprechers, welche aus der Umwandlung von Schallwellen mit einem Mikrofon entsteht [10, S. 25]. Woraus das Sprachsignal besteht und welche Faktoren Einfluss darauf haben, wird im Folgenden näher beschrieben.

Bestandteile eines Sprachsignals

Ein Sprachsignal besteht im Idealfall nur aus der zu übertragenden Information. Tatsächlich haben jedoch einige Störfaktoren Einfluss auf dessen Qualität, welche in Abbildung 2.1 dargestellt sind. Anhand der Abbildung ist ersichtlich, dass sowohl beim Ursprung der Aussage als auch bei der Übertragung und deren späterer Umwandlung störende Einflüsse auf Sprachsignale einwirken.

Diese Einflüsse können durchaus positiv sein, wenn nicht nur die gesprochene Aussage für einen bestimmten Anwendungsfall von Interesse ist. Bei der Sprechererkennung werden beispielsweise einige dieser Einflüsse herangezogen, um einen Sprecher identifizieren zu können [10, S. 29]. Dazu zählen vor allem die Sprechgewohnheiten und die körperlichen Gegebenheiten des Sprechers, die in der Abbildung als Physiologie bezeichnet werden.

Da dies jedoch nicht für die vorliegende Arbeit relevant ist, wird nicht näher auf diese Anwendungen der Computerlinguistik eingegangen.

2.1.3 Methoden

Bei der Spracherkennung unterscheidet man mehrere Methoden, die sich in deren Funktionsweise grundlegend unterscheiden. Zwei dieser Methoden werden nachfolgend näher beschrieben: zum einen die Methode des Mustervergleichs und zum anderen die statistische Spracherkennung.

Mustervergleich

Die Grundlage dieser Methode ist eine Datenbank, die alle zu erkennenden Wörter als Sprachmuster enthält. Diese gespeicherten Sprachmuster werden dann mit dem Sprachmuster des zu erkennenden Sprachsignals verglichen. Man misst *„die Verschiedenheit von zwei Sprachsignalen“*, die auch als Distanz bezeichnet wird [10, S. 305].

Eigenschaften von Sprachmustern Die wichtigste Eigenschaft von guten, sprecherunabhängigen Sprachmustern ist eine möglichst geringe Distanz bei gleichbedeutenden Wörtern und eine möglichst große bei unterschiedlichen Wörtern. Dazu ist es nötig, einen Algorithmus zu verwenden, der Sprachmuster generiert, die möglichst unabhängig vom Sprecher sind. Dazu müssen die in Abbildung 2.1 gezeigten Einflüsse entsprechend gefiltert oder dürfen bei der Erstellung nicht berücksichtigt werden [10, S. 305].

Erstellung von Sprachmustern Zur Erstellung von Sprachmustern werden die zu erkennenden Wörter von mehreren verschiedenen Sprechern aufgenommen und weiterverarbeitet. Grundsätzlich wäre es möglich, ein Sprachmuster aus einer einzelnen Aufnahme zu generieren. Da es jedoch praktisch keinen Algorithmus gibt, der völlig sprecherunabhängige Sprachmuster generiert, wären diese zu sehr von diesem einen Sprecher beeinflusst, was die Erkennbarkeit der Wörter bei anderen Sprechern erschwert.

Anwendung Der Mustervergleich wird verwendet, wenn einzelne Wörter, kurze Sätze oder Ziffern bzw. Zahlen erkannt werden sollen. Allgemein betrachtet immer dann, wenn die zu erkennende Sprachgrammatik nicht umfangreich ist [10].

Statistische Spracherkennung

Bei der statistischen Spracherkennung werden Methoden der Statistik bzw. der Wahrscheinlichkeitsrechnung verwendet, um ein Sprachsignal in eine Folge von Wörtern umzuwandeln. Diese Problemstellung kann *„aus informationstheoretischer Sicht [...] als Decodierungsproblem betrachtet werden“* [10, S. 327]. Wie in Abbildung 2.2 dargestellt, wird aus dem Sprachsignal eine *„Merkmalssequenz X extrahiert“*, aus der ein linguistischer Decoder die gesprochene Wortfolge schätzt [10, S. 327]. „W“ steht hierbei für die Wortfolge,

die der Sprecher über seinen Sprechapparat ausspricht, „s“ für das dabei entstehende akustische Signal und $\hat{W}$ für die vom Spracherkenner geschätzte Wortfolge.

Das Ziel ist es, *„aus der Merkmalssequenz* $\boldsymbol{X}$ *eine möglichst gute Schätzung* $\hat{W}$ *der geäusserten [sic] Wortfolge zu ermitteln“* [10, S. 327].

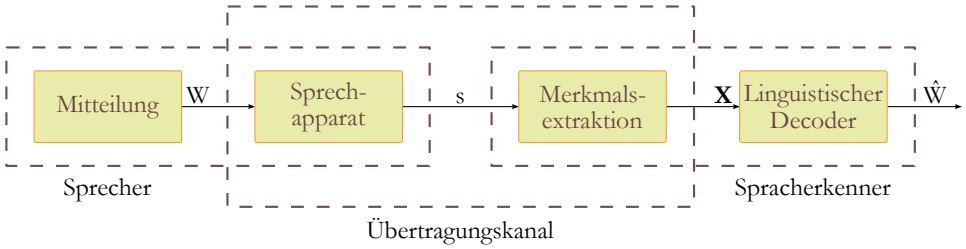


Abbildung 2.2: Informationstheoretische Sicht der Spracherkennung [10, S. 327]

2.1.4 Sprachmodelle

Allgemein betrachtet wird das Sprachmodell als *„eine Sammlung von ‘A-priori-Kenntnissen über die Sprache bezeichnet“* [10, S. 369]. Diese Sammlung besteht aus *„eine/r/ Menge von Wortfolgen ohne bestimmte akustische Realisierung“* und wird aufgrund von Erfahrungswerten im Voraus berechnet [10, S. 369]. Basierend auf Erfahrungswerten über eine Sprache wird hierbei hauptsächlich die Häufigkeit berechnet, in der Wörter in bestimmten Situationen verwendet werden. Unterschieden wird in diesem Zusammenhang zwischen zwei Modellierungsmethoden [10, S. 369f]:

- **Statistische Sprachmodelle**: Die Erfahrungswerte werden durch Zählen oder Messen ermittelt, wie beispielsweise durch Ermitteln der Häufigkeit von Wörtern. Als Sprachmodell wird hier die Wahrscheinlichkeitsverteilung bezeichnet, die mittels „Hidden Markov Models“ beschrieben werden kann [10, S. 370]. Statistische Sprachmodelle werden teilweise auch zur Grammatik eines Spracherkennungssystems gezählt [2, S. 20].
- **Wissensbasierte Sprachmodelle**: Die Erfahrungswerte werden durch Gesetzmäßigkeiten einer Sprache, wie z.B. deren grammatikalischen Zusammenhänge ermittelt.

2.1.5 Grammatik

Cohen definiert die Grammatik eines Spracherkennungssystems wie folgt [2, S. 21]:

> *„The grammar is the definition of all the things the caller can say to the system and be understood. It includes a definition of all the possible strings of words the recognizer can handle, along with the rules for associating a meaning with those strings (e.g. by filling slots).“*

Während diese Definition sehr anwendungsorientert formuliert ist, basieren Grammatiken auf der Theorie der formalen Sprachen. Nach Noam Chomsky, dem *„Begründer der formalen Sprachen“* werden Grammatiken in vier Klassen eingeteilt [10, S. 139f]. Die Einteilung erfolgt hierbei anhand der Komplexität der Grammatiken, wobei die am stärksten eingeschränkte Grammatik die sogenannte „Typ-3-“ oder „reguläre Grammatik“ ist [10, S. 142f].

Die in der Praxis am häufigsten verwendete Art von Grammatiken ist die sogenannte „kontextfreie Grammatik“, die nach Chomsky auch als „Typ-2-Grammatik“ [10, S. 142] bezeichnet und im Folgenden näher erläutert wird.

Kontextfreie Grammatik

Kontextfreie Grammatik wird auch als „Backus-Naur Form“ (BNF) bezeichnet und besteht im Wesentlichen aus den vier nachfolgend angeführten Komponenten[6, S. 27]. Ein Beispiel einer kontextfreien Grammatik für eine sehr einfache Spracherkennung ist in Listing 2.1 ersichtlich.

Terminalsymbole sind die erlaubten Wörter und Zeichen, die auch als Token bezeichnet werden. In Listing 2.1 wären dies die Wörter „Hi“, „Hallo“, „Max“ und „Erika“.

Nichtterminalsymbole sind Konstrukte, die sich aus Terminal- und Nichtterminalsymbolen zusammensetzen. Bezogen auf Listing 2.1 sind das die Konstrukte „Grußwort“, „Name“ sowie „Grußformel“, wobei sich „Grußwort“ aus zwei Terminal- und „Grußformel“ aus zwei Nichtterminalsymbolen zusammensetzt.

Produktionen sind die einzelnen Regeln, die auf deren linker Seite ein Nichtterminalsymbol haben und auf der rechten Seite eine Kombination aus Terminal- und Nichtterminalsymbolen. Listing 2.1 besteht aus drei Produktionen, die den einzelnen Zeilen entsprechen.

Darstellungsform Zur Darstellung von Grammatiken gibt es sowohl schriftliche als auch grafische Möglichkeiten. Schriftlich werden Grammatiken meist in BNF oder EBNF Form dargestellt, welche auch als Grundlage zum häufig verwendeten „Java Speech Grammar Format“ (JSGF) dient. Grafisch erfolgt

die Darstellung in sogenannten Syntaxdiagrammen. Listing 2.1 zeigt ein einfaches Beispiel in BNF Form. Die entsprechende grafische Repräsentation der daraus resultierenden Grammatik ist in Abbildung 2.3 dargestellt. Die in einem Spracherkennungssystem erkennbaren Sätze werden in der schriftlichen Darstellung durch das Einsetzen der Terminalsymbole an den entsprechenden Stellen der Produktionsregel gebildet. Aus der grafischen Darstellung können alle Wortpaare durch das Lesen der Grafik in Richtung der abgebildeten Pfeile gebildet werden. Sowohl die schriftliche als auch die grafische Darstellung des beschriebenen Beispiels ergeben die folgenden vier Wortpaare:

- Hi Max
- Hallo Max
- Hi Erika
- Hallo Erika

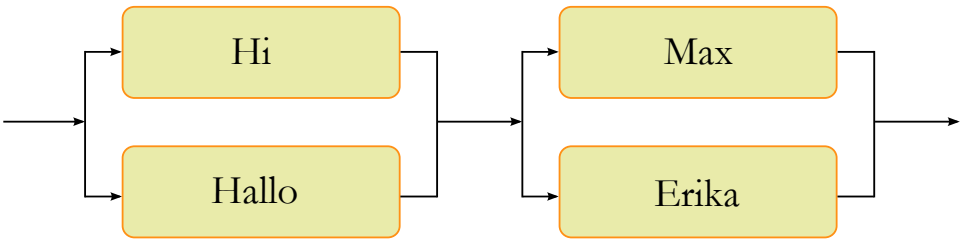


Abbildung 2.3: Ablaufdiagramm der Grammatik aus Listing 2.1

Listing 2.1: Beispiel einer kontextfreien Grammatik in BNF-Darstellung

```
1 Grußwort = "Hi" | "Hallo" ;
2 Name = "Max" | "Erika" ;
3 Grußformel = Grusswort Name ;
```

Listing 2.2: JSGF-Version der Grammatik „Grüßen“ aus Listing 2.1

```
1  #JSGF V1.0;
2  grammar gruessen;
3
4  <grusswort> = hi | hallo;
5  <name> = max | erika;
6
7  <grussformel> = <grusswort> <name>;
```

Zur schriftlichen Darstellung einer Grammatik werden verschiedene Formate verwendet. Für formale Sprachen werden meist BNF oder darauf aufbauende Formate wie „Extended BNF“, „Augmented BNF“ oder auch JSGF verwendet. Da sie auf der gleichen Logik basieren, können Grammatiken zwischen unterschiedlichen Formaten konvertiert werden. Dies veranschaulicht die in den Listings 2.1 und 2.2 angeführten Grammatiken.

Die in Listing 2.2 angeführte Grammatik stellt ein sehr vereinfachtes Beispiel dar. Mithilfe von JSGF können auch sehr komplexe Grammatiken erstellt werden.

Beispielsweise können durch die an der Programmiersprache Java angelehnte Paketstruktur JSGF-Grammatiken modular aufgebaut und sehr einfach wiederverwendet werden. Eine JSGF-Datei entspricht dabei einer Grammatik, die eine oder mehrere Regeln enthalten kann. Durch das Importieren von Grammatiken ist es möglich, als „public“ gekennzeichnete Regeln in anderen Grammatiken zu verwenden.

Weiters können bestimmte Teile einer Regel mit einem „*“ oder „+“ gekennzeichnet werden. Dies bedeutet, dass einzelne Teile „0 bis beliebig“ bzw. „1 bis beliebig“ oft im Sprachsignal vorkommen können. Das entspricht den in regulären Ausdrücken verwendeten Quantoren [43]. Zusätzlich ist es möglich, mehrere Wörter mit Klammern zu gruppieren oder mit eckigen Klammern als optional zu kennzeichnen.

Einige Beispiele für JSGF-Grammatiken sind in [24, Abschnitt 5] verfügbar.

2.1.6 Google-Spracherkennung

Über die im Masterprojekt verwendete serverseitige Spracherkennung der Firma Google sind öffentlich keine wissenschaftlichen Informationen verfügbar. Eine entsprechende Anfrage an Google wurde nicht beantwortet.

Google verfügt über eine sehr große Menge an Daten in schriftlicher Form, die über Dienste wie die Google-Suche oder Google Mail gesammelt und verarbeitet werden. Aufgrund dieser Tatsache ist anzunehmen, dass der Spracherkennung statistische und wissensbasierte Modelle zu Grunde liegen, die aus dieser Datenmenge berechnet werden.

Ein Online-Artikel des „Handelsblatt“ enthält einige Informationen über die Ziele und Visionen, die Google zum Thema Sprachverarbeitung hat. So wird das langfristige Ziel wie folgt beschrieben: „Jedes gesprochene Wort muss am Ende praktisch ohne Verzögerung erkannt, blitzschnell korrekt übersetzt und am anderen Ende fehlerfrei betont in einer anderen Sprache wieder ausgegeben werden“ [31].

Die minimale Anzahl an Sprachmustern für eine bestimmte Sprache wird mit 250.000 beziffert. Um Sprachbeispiele zu sammeln, beschäftigt das Unternehmen Personal, das rund um die Welt nur dieser Aufgabe nachgeht. Zusätzlich werden Sprachbeispiele von Benutzern der Spracherkennung des Smartphone-Betriebssystems Android sowie der im Google Chrome Browser integrierten Spracherkennung verwendet.

2.2 Sprachein- und Sprachausgabe im Web

Dieser Abschnitt soll einen kurzen Überblick über die Möglichkeiten der Sprachausgabe sowie der Spracheingabe in Browser-Umgebungen geben.

2.2.1 Sprachausgabe

Unter Sprachausgabe im Internet versteht man allgemein den Vorgang des Vorlesens einer Internetseite. Dies wird vor allem von blinden oder sehschwachen Menschen verwendet, um Internet-Inhalte verwenden zu können. Zu diesem Zweck wird am Rechner des Benutzers eine „Screen-Reader-Software“ installiert, die die Inhalte des Computers als Sprache oder auch über eine Braille-Zeile[1]ausgibt. Da der Fokus dieser Arbeit auf der Spracheingabe liegt, wird an dieser Stelle nicht näher auf dieses Thema eingegangen.

2.2.2 Spracheingabe

Für den Zugriff auf ein Mikrofon innerhalb eines Browsers existieren verschiedene Technologien.

Java Applets

Java bietet mit sogenannten „Applets“ die Möglichkeit, Programme auf dem Rechner des Benutzers auszuführen. Dabei wird das Programm zuvor zum Benutzer übertragen und dann ausgeführt. Zum Starten von Applets muss auf dem ausführenden Rechner eine „Java Virtual Machine“ installiert sein, die dann innerhalb eines Java-fähigen Browsers ausgeführt wird [13].

Für den Zugriff auf das Mikrofon stehen verschiedene Klassen und Funktionen bereit. Die aufgenommenen Daten können anschließend direkt mit einer Spracherkennungs-Bibliothek wie z.B. Sphinx-4 [11] weiterverarbeitet oder an einen Server übertragen werden.

Adobe Flash

Ähnlich wie Java Applets werden Adobe Flash Anwendungen zuerst zum Benutzer übertragen und dann von einem Plug-In im Browser ausgeführt. Die in ActionScript geschriebenen Anwendungen haben sowohl Zugriff auf das Mikrofon als auch auf die Webcam des Benutzers. Bevor diese Geräte jedoch verwendet werden können, muss der Benutzer die Erlaubnis dazu erteilen. Danach können die Daten ebenso wie mit Java direkt verwendet oder an einen Server zur Weiteren Bearbeitung gesendet werden.

[1]Eine Braille-Zeile ist ein Gerät, das Texte zeilenweise als Blindenschrift ausgibt. [41]

HTML5

Aktuell existieren Entwürfe eines W3C-Standards für einen nativen Zugriff auf das Mikrofon und die Webcam eines Benutzers. Die grundsätzliche Idee besteht darin, eine JavaScript-Schnittstelle zu definieren, mit der es möglich ist, auf entsprechende Geräte des Computers zuzugreifen.

Die Kernkomponente dieser geplanten Schnittstelle ist die „getUserMedia"-Funktion, welche im W3C-Entwurf [16] näher beschrieben ist. Diese Funktion kann dazu benutzt werden, entweder auf Audio- oder Videogeräte des Benutzers zuzugreifen. Dabei kann über Optionen gesteuert werden, auf welche dieser Geräte konkret verwendet werden sollen. Dabei ist immer die aktive Freigabe des Benutzers erforderlich [16, Abschnitt 2.1.2].

Speech Input API

Ein Entwurf für einen W3C-Standard für derartige Eingabefelder und einer API namens „Speech Input API" wurde bereits Ende 2010 von Google eingebracht [21]. Zum aktuellen Zeitpunkt ist diese Schnittstelle nur im Google Chrome Browser verfügbar.

WebRTC

Aufbauend auf der beschriebenen „getUserMedia"-Funktion wird an einer Schnittstelle namens WebRTC[2]zur direkten Kommunikation zwischen zwei oder mehreren Browsern gearbeitet. Der Google Chrome-Browser bietet eine experimentelle Implementierung der Schnittstelle, die in den sogenannten „Developer-Builds" enthalten ist [22].

2.3 Barrierefreiheit im Internet

Allgemein betrachtet, bedeutet Barrierefreiheit (engl. „accessibility"), *„dass Gegenstände, Medien und Einrichtungen so gestaltet werden, dass sie von jedem Menschen unabhängig von einer eventuell vorhandenen Behinderung uneingeschränkt benutzt werden können."* [40]. Während dieses Thema im Alltag beispielsweise im Bereich des barrierefreien Zugangs zu Gebäuden bereits sehr weit verbreitet ist, wird es bei der Entwicklung von Internetseiten häufig vernachlässigt. Dieser Abschnitt soll eine Einführung in diese Thematik geben und einige Richtlinien und Frameworks beschreiben, die zur Verbesserung der Barrierefreiheit im Internet dienen können.

2.3.1 Warum Barrierefreiheit im Internet wichtig ist

Das Internet ist eine Sammlung an Inhalten, die weltweit abgefragt werden kann. Dazu zählen sowohl Texte als auch Musikstücke, Fotos und Videos.

[2]Die Abkürzung 'WebRTC' steht für „Web Real-Time Communication"[15]

Da der größte Teil der Inhalte im Internet visueller Natur ist, sind es insbesondere Sehbeeinträchtigungen, die Probleme beim Konsumieren der Inhalte bereiten. Durch die steigende Popularität von Videoportalen wie „YouTube“ müssen in Zukunft auch vermehrt Lösungen für hörbeeinträchtigte Menschen gefunden werden. Der Grund, warum Internetangebote barrierefrei gestaltet werden sollen, beschreibt das folgende Zitat von einem blinden Internetbenutzer: *„For me being online is everything. Its [sic] my hi-fi, my source of income, my supermarket, my telephone. Its [sic] my way in.“* [5, S. 2].

In der Einleitung zu den „Web Content Accessibility Guidelines 1.0“ werden zur Verdeutlichung der Problematik einige Punkte angeführt, die beschreiben, welche Probleme Benutzer von Internetseiten haben können [18, Kapitel 1]:

- *„They may not be able to see, hear, move, or may not be able to process some types of information easily or at all.“*
- *“They may have difficulty reading or comprehending text.“*
- *“They may not have or be able to use a keyboard or mouse.“*
- *“They may have a text-only screen, a small screen, or a slow Internet connection.“*
- *“They may not speak or understand fluently the language in which the document is written.“*
- *“They may be in a situation where their eyes, ears, or hands are busy or interfered with (e.g., driving to work, working in a loud environment, etc.).“*
- *“They may have an early version of a browser, a different browser entirely, a voice browser, or a different operating system.“*

Diese Auflistung macht deutlich, welche Erwägungen in die Entwicklung von Internetseiten einfließen sollen. Einige der Richtlinien zur Verbesserung der Barrierefreiheit im Internet sowie deren Entwicklung werden im folgenden Abschnitt erläutert. Diese sollen den Betreibern von Internetseiten bei der Gestaltung von barrierefreien Internetseiten helfen.

2.3.2 Web Content Accessibility Guidelines 1.0 (WCAG1)

Die erste Richtlinie des W3C zum Thema Web-Accessibility ist die WCAG 1.0 (manchmal auch als „WCAG1“ abgekürzt), die seit 1999 den Status einer W3C-Empfehlung hat. Sie richtet sich sowohl an Betreiber und Entwickler von Internetseiten, als auch an die Entwickler von sogenannten „authoring tools“ wie HTML-Editoren [18, Abschnitt 1]. Insgesamt beinhaltet die WCAG 1.0 eine Reihe von kleineren Richtlinien, die jeweils eine Beschreibung

sowie eine Checkliste inkl. der Angabe einer Priorität von eins (verpflichtend) bis drei (optional) umfassen. Nachfolgend werden die aus Sicht dieser Arbeit wichtigsten Richtlinien beschrieben. Da es sich hierbei um einen grundlegenden Leitfaden zum sinnvollen Einsatz von HTML handelt, werden diese entsprechend detailliert beschrieben. Die vollständige Liste ist unter [18] verfügbar.

Alternativen für bestimmte Inhalte bereitstellen

Ziel dieser Richtlinie, die im englischen Original *„Provide equivalent alternatives to auditory and visual content“* [18, Abschnitt 6.1] lautet, ist die Bereitstellung von Alternativen für nicht jedem Benutzer zugänglichzen Inhalten. Explizit genannt wird hierbei das sogenannte „alt“-Attribut der HTML-Tags „img“, „input“ und „applet“ genannt, das eine schriftliche Repräsentation des gezeigten Inhaltes enthalten soll.

Die Formulierung dieses alternativen Textes hängt vor allem von der Verwendung der Inhalte ab. In [18, Abschnitt 1] wird als Beispiel ein aus dem Weltall aufgenommenes Bild der Erde auf einer Internetseite angeführt. Wenn dieses Bild rein zur Dekoration dient, würde der Text *„Photograph [sic] of the Earth as seen from outer space“* genügen. Wenn das Bild jedoch als Link dient, um mehr Informationen zum Planeten Erde anzuzeigen, könnte der Text *„Information [sic] about the Earth“* verwendet werden, um den Benutzer dazu aufzufordern, auf das Bild zu klicken. Für komplexere Inhalte, die schwer bzw. gar nicht als Text formuliert werden können, wird die Verwendung eines sogenannten „Description-Link“ empfohlen, der zu einer umfassenderen Beschreibung des Inhaltes führt [18].

Neben dem bereits erwähntem „alt“-Attribut enthält die Checkliste unter anderem die folgenden Punkte, welche beide Priorität eins besitzen und somit verpflichtend umzusetzen sind, wenn die Internetseite WCAG1-konform sein soll:

- Bereitstellen einer Audio-Beschreibung mit den wichtigsten Inhalten von Video-Inhalten [18, Checkpoint 1.3].
- Synchronisieren der Textbeschreibungen von zeitbasierten Inhalten wie Videos, Animationen oder Präsentationen, um den jeweils aktuell gezeigten Inhalt zu beschreiben [18, Checkpoint 1.4].

Während der Inhalt des „alt“-Attributes eine kurze und prägnante Beschreibung des Zwecks eines Elementes geben soll, kann das „longdesc“-Attribut verwendet werden, um eine längere Beschreibung von bestimmten Inhalten zu bieten.

Richtiges Verwenden von „Markup“ und „Stylesheets“

Die Grundaussage dieser Richtlinie ist es, Markup und Stylesheets für den jeweils richtigen Zweck zu verwenden. Als Beispiel für Markup werden hier-

bei die Verwendung von Tabellen zu Layout-Zwecken und jene von Überschriften zur Änderung der Schriftgröße genannt. Umgekehrt sollen Elemente wie „pre“ oder „div“ nicht zur Gestaltung von Tabellen verwendet werden. Präsentations-Elemente wie „font“ zur Beeinflussung der Schriftgröße sollen grundsätzlich gemieden werden [18, Checkpoint 3.3], da eine HTML-Seite die Struktur definiert, Stylesheets die Präsentation festlegt und diese beiden Bereiche getrennt werden sollten.

Weiters wird zum Verwenden von Überschriften wie „h1“ und „h2“ zum Strukturieren von Inhalten geraten. Ferner sollen diese korrekt geschachtelt werden, „h2“ also immer als Unterabschnitt von „h1“ [18, Checkpoint 3.5].

Deklarieren unterschiedlicher Sprachen auf einer Seite

Richtlinie 4 beschreibt die Wichtigkeit, sowohl die für den Hauptinhalt der Internetseite, als auch für anderssprachige Inhalte die jeweils verwendete Sprache anzugeben. Während die hauptsächlich verwendete Sprache in einem „HTTP Header“ angegeben werden kann, bietet die HTML4 Spezifikation die Möglichkeit, mit dem „lang“-Attribut die Sprache des in Container-Elementen enthaltenen Inhalts anzugeben [32, Abschnitt 8.1].

Verwenden von W3C-Technologien

Diese Richtlinie zielt darauf ab, dass für die Entwicklung von Internetseiten nur W3C-Technologien wie beispielsweise HTML, CSS und JavaScript verwendet werden sollen. Diese Empfehlung rührt daher, dass Inhalte wie Adobe Flash-Applikationen oder PDF-Dateien immer ein Plug-In oder ein externes Programm benötigen, um verwendet werden zu können. Im Sinne der Barrierefreiheit wird daher empfohlen, für alle Inhalte, die nicht mit W3C-Technologien realisiert oder konsumiert werden können, eine gleichwertige Alternative anzubieten [18, Guideline 11]. Weitere Checkpoints umfassen unter anderem:

- Nicht mehr unterstützte Bestandteile („deprecated features„) wie das HTML „font“-Element sollen nicht verwendet werden [18, Checkpoint 11.2].
- Inhalte sollen in verschiedenen Formen, wie beispielsweise in unterschiedlichen Sprachen vorhanden sein. Dazu sollen die Spracheinstellungen des Benutzers berücksichtigt, immer jedoch eine Möglichkeit geboten werden, diese zu ändern [18, Checkpoint 11.3].

Klare Navigations-Mechanismen anbieten

Es wird empfohlen, eine klare und konsistente Navigation durch die einzelnen Seiten eines Internetauftritts anzubieten [18, Checkpoint 13.4]. Das Ziel des Links soll durch den Link-Text klar beschrieben und auch ohne den Kontext

der Seite nachvollziehbar sein, was in der Richtlinie wie folgt beschrieben wird: *„Link text should be meaningful enough to make sense when read out of context - either on its own or as part of a sequence of links.“* [18, Checkpoint 13.1].

2.3.3 Web Content Accessibility Guidelines 2.0 (WCAG2)

Ähnlich wie WCAG1 (vgl. Abschnitt 2.3.2) sind die WCAG 2.0 eine Sammlung von Richtlinien zur Verbesserung der Barrierefreiheit von Internetseiten. Im Unterschied zu den WCAG 1.0 ist die Version 2.0 technologie-unabhängig (engl. „technology-neutral“ [27, 4:00]) und definiert allgemeine Grundsätze für ein barrierefreies Internet. Die Richtlinien sind in vier Bereiche unterteilt, wobei für jede Richtlinie Hinweise zur Umsetzung gegeben werden[17]:

- *„Perceivable - Information and user interface components must be presentable to users in ways they can perceive.“*
- *„Operable - User interface components and navigation must be operable.“*
- *„Understandable - Information and the operation of user interface must be understandable.“*
- *„Robust - Inhalte müssen robust genug sein, damit sie zuverlässig von einer großen Auswahl an Benutzeragenten einschließlich assistierender Techniken interpretiert werden können.“*

Da die Veränderungen zwischen den beiden Versionen aus technischer Sicht sehr gering sind, wird die Version 2.0 nicht näher betrachtet. Weitere Informationen sind in [17] zu finden.

2.3.4 WAI-ARIA

Die „Accessible Rich Internet Applications Suite“ (ARIA) der „Web Accessibility Initiative“ (WAI) ist eine Erweiterung von HTML bzw. XHTML, die unter anderem Rollen und Zustände definiert, die es beispielsweise Screen-Reader Programmen ermöglicht, dynamische Web-Applikationen besser zugänglich zu machen. Das Ziel ist die Anreicherung von HTML-Elementen mit semantischen Meta-Informationen, die in der „WAI-ARIA Taxonomy“ definiert sind. Die Taxonomie ist sowohl als Grafik als auch als RDF-Datei verfügbar und definiert die Zusammenhänge der verschiedenen Rollen und Zustände, die in WAI-ARIA verwendet werden [39].

Rollen

Rollen werden in WAI-ARIA in vier Gruppen eingeteilt, wobei die Gruppe der abstrakten Rollen nicht in Internetseiten verwendet werden darf und nur in der Taxonomie enthalten ist. Einige Rollen, die für Widgets verwendet werden, sind in Tabelle 2.1 aufgelistet. Die Tabelle enthält dabei die Rollen

von Elementen sowie jene der Container-Elemente, in denen diese zusammengefasst werden können.

Die Rollen der „Document Structure“-Gruppe, aufgelistet in [30, Abschnitt 5.3.3], sind nur bedingt relevant für die vorliegende Arbeit. Einzig die Rolle „article“ könnte ähnlich dem gleichnamigen HTML5-Element zur Bestimmung des Kontextes eines Elementes herangezogen werden (vgl. Abschnitt 2.5.2).

Rolle	**Container**	**Beschreibung**
tab	tablist	Beschreibt eine Komponente, die aus mehreren Tabs bestehen kann.
menuitem	menubar oder menu	Beschreibt eine Menüleiste oder ein Menü
button	Keiner	Beschreibt einen Button, der nicht mit den üblichen HTML-Elementen definiert ist

Tabelle 2.1: Widget-Rollen aus WAI-ARIA [30, Abschnitt 5.3.2]

Die vierte Gruppe beschreibt die sogenannten „Landmark-Roles“, deren relevante Rollen in Tabelle 2.2 aufgelistet sind. Diese werden zur Auszeichnung von Seitenbereichen verwendet, die einem bestimmten Zweck dienen. Dazu zählen vor allem die Seiten-Navigation und die Suchfunktion.

Rolle	**Beschreibung**
navigation	Beschreibt eine Navigationsleiste
search	Beschreibt ein oder mehrere Elemente, die zur Suche auf der Seite dienen
banner	Beschreibt einen Banner, welcher das Seitenlogo oder den Seitensponsor enthält.

Tabelle 2.2: „Landmark“ Rollen aus WAI-ARIA [30, Abschnitt 5.3.4]

Zustände Um die veränderlichen Zustände bestimmter HTML-Elemente zu definieren, bietet die WAI-ARIA Spezifikation eine Reihe von Attributen zur genauen Beschreibung des jeweils aktuellen Zustands eines Elementes. Einige dieser Attribute sind in Tabelle 2.3 aufgelistet. Zustände wie „aria-checked“ oder „aria-selected“ können dabei sowohl für Formular-Elemente wie Checkboxen oder Radiobuttons, als auch für Widgets verwendet werden, die z.B. aus „DIV“-Containern bestehen.

Für die Sprachsteuerung von Internetseiten haben die meisten Zustände keine Verwendung. Lediglich der „aria-hidden“-Zustand könnte verwendet werden, um nicht sichtbare Elemente nicht für die Sprachsteuerung zuzulassen.

Attribut	Beschreibung
aria-checked	Element ist aktiviert
aria-hidden	Element ist nicht sichtbar
aria-selected	Element ist ausgewählt
aria-grabbed	Element ist gerade in einer „Drag-and-Drop“-Operation aktiv

Tabelle 2.3: Beispiele für WAI-ARIA-Zustände [30]

Relationen Mehr relevante Informationen lassen sich über die ebenfalls durch Attribute ausgedrückten Relationen zwischen Elementen sammeln. In Tabelle 2.4 sind die für die vorliegende Arbeit relevanten Attribute aufgelistet.

Attribut	Beschreibung
aria-activedescendant	Identifiziert das momentan aktive Kind-Element
aria-label	Definiert die Beschreibung eines Elementes
aria-labelledby	Identifiziert ein Element, welches das aktuelle Element beschreibt

Tabelle 2.4: Beispiele für WAI-ARIA-Relationen [30]

Identifizierenden Attributen wird eine HTML-ID zugewiesen, die ein Element der gleichen Internetseite besitzt. Das Attribut „aria-activedescendant“ kann beispielsweise in sogenannten „Carousel-Widgets“ verwendet werden, die je eines aus mehreren Elementen sichtbar schalten. Der Wert des Attributes ist in diesem Fall die ID des jeweils sichtbaren Elementes.

Die Attribute „aria-label“ und „aria-labelledby“ können dazu verwendet werden, zusätzliche Informationen zu einem Element zu extrahieren und diese für Sprachkommandos einzusetzen.

Live Regions

Die WAI-ARIA Spezifikation führt das Konzept der „Live Regions“ ein, um Veränderungen im DOM-Baum („dom mutations“) bestimmter Regionen einer Internetseite überwachen zu können. Dazu wird das Attribut „aria-live“ eingeführt, das beispielsweise einem „DIV“-Container zugewiesen werden kann.

Die Veränderungen im DOM-Baum des Elementes können mit JavaScript überwacht werden (vgl. Abschnitt 5.6). Der Vorteil in der Verwendung von Live-Regions ist, dass nicht der gesamte DOM-Baum auf Veränderungen überwacht werden muss und die Internetseite bzw. Web-Anwendung weniger rechenintensiv ist.

2.4 Volltextsuche

Eine Volltextsuche wird verwendet, um anhand von Stichwörtern oder Sätzen aus einer Menge von Dokumenten die passendsten Ergebnisse zu finden. Im Englischen wird Volltextsuche oft auch als „information retrieval“ bezeichnet [9, S. 6]. Die am weitesten verbreitete Lösung dieser Art bietet das „Apache Lucene“-Projekt, welches nachfolgend näher beschrieben wird.

2.4.1 Apache Lucene

Apache Lucene ist eine Bibliothek, die in der originalen Fassung in Java geschrieben, in weiterer Folge jedoch mit weiteren Programmiersprachen wie z.B. C#, PHP oder Ruby umgesetzt wurde [9, S. 3]. Lucene ist ein Open-Source Projekt, das von der „Apache Software Foundation“ unter der gleichnamigen „Apache Software Lizenz“ veröffentlicht wird. Die Bibliothek bietet eine einfach zu verwendende Schnittstelle an, die ohne tiefgreifendes Wissen über die darunter liegenden Mechanismen in Applikationen eingebaut werden kann. Lucene wird sowohl als eigenständige Bibliothek in Anwendungen als auch als Grundlage bzw. Kerntechnologie für spezialisierte Anwendungen verwendet. Zwei dieser Projekte sind [9, S. 12]:

- **Apache Solr** ist ein Suchserver, der als Grundlage Lucene verwendet.
- **Nutch** ist eine hoch skalierbare Internet Suchmaschine, die aus einem Web-Crawler und einer Suchmaschine besteht. Aus Nutch entstand auch das „Apache Hadoop Framework“[3], das es ermöglicht *„intensive Rechenprozesse mit großen Datenmengen [...] auf Computerclustern durchzuführen“* [42].

Funktionsweise

Wie in Abbildung 2.4 dargestellt, ist der „Index“ die zentrale Komponente einer Lucene-Anwendung. Dieser beinhaltet die Informationen, die von der Bibliothek benötigt werden, um anhand von Suchanfragen des Benutzers die treffendsten Ergebnisse zu finden. Neben dem Administrations- und Analytics-Interface kann die Funktionsweise in zwei Bereiche geteilt werden: dem Hinzufügen von neuen Dokumenten in den Suchindex sowie der Bearbeitung von Suchabfragen. Nachdem einige in Abbildung 2.4 vorkommende Begriffe definiert werden, werden diese beiden Bereiche und deren Aufgaben beschrieben.

Begriffsdefinitionen Zum Verständnis der Funktionsweise von Apache Lucene müssen zuerst einige Begriffe definiert werden.

[3] Mehr Informationen unter: http://hadoop.apache.org

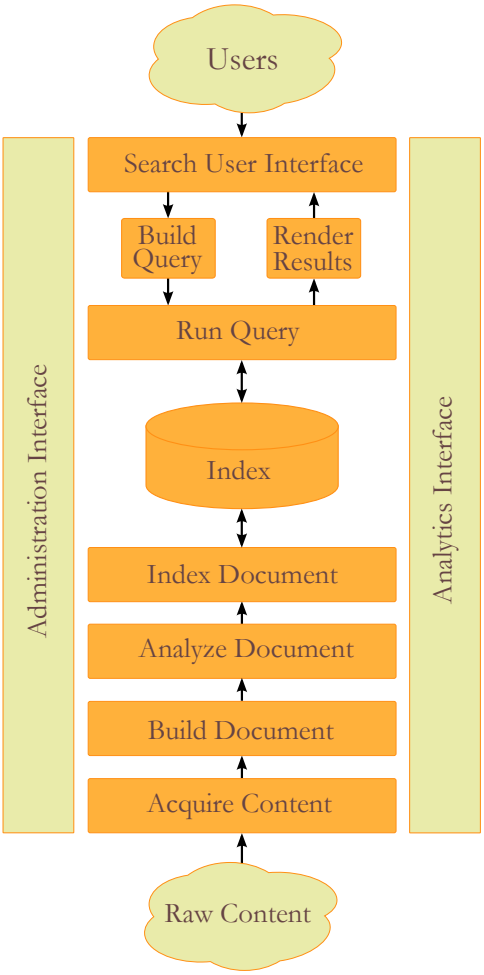


Abbildung 2.4: Funktionsweise von Apache Lucene [9, S. 10]

Content Der Content beschreibt jegliche Daten, die in ungeordneter Form vorliegen. Da Lucene eine Bibliothek zur Textsuche ist, handelt es sich beim Content meist um Textdaten. Möglich sind jedoch beispielsweise auch Audio- oder Videodateien, aus denen die enthaltenen Meta-Informationen indiziert werden können.

Document Als Dokument beschreibt man kleinere Einheiten von Informationen, die aus den Rohdaten extrahiert wurden. Ein Dokument kann beispielsweise eine E-Mail oder eine Internetseite sein und besitzt ein oder mehrere benannte Felder, die sinnvoll mit Werten befüllt werden müssen [9, S. 12].

Index Ein Suchindex beinhaltet alle Informationen, die von der Such-Engine benötigt wird. Der Index wird über eine definierte Schnittstelle mit Daten befüllt, welche anschließend analysiert und in geeigneter Form abgespeichert werden. Nur Informationen, die im Index enthalten sind, können von der Suche gefunden werden.

Query Eine Query (engl. für Abfrage) ist ein Befehl, der in einer festgelegten Abfragesprache geschrieben wurde. In der Abfrage wird festgelegt wonach gesucht wird und wie diese Suchergebnisse gefiltert oder sortiert werden sollen.

Befüllen des Index Bezogen auf Abbildung 2.4 besteht das Befüllen des Index aus vier Schritten, die den „Raw Content“ verarbeiten und in den Index laden.

Aquire Content [9, S. 11] Da die Daten zur Verarbeitung aus verschiedenen Quellen wie beispielsweise Textdateien oder relationalen Datenbanken kommen können, wird eine Komponente benötigt, die entsprechend den Zugriff auf diese Quellen realisiert. Neben diesen direkten Zugriffsmethoden kann diese Komponente auch ein sogenannter „Web Crawler“ sein, der Informationen von Internetseiten extrahiert. Wichtig in diesem Schritt ist, insbesondere bei sehr großen Datenmengen die Effizienz der Komponente. Diese kann u.a. durch die inkrementelle Verarbeitung von veränderten Inhalten erreicht werden.

Die Lucene-Bibliothek bietet keine Funktionalität für diesen Schritt an, weswegen er von der verwendeten Applikation implementiert oder einer bestehenden Software-Komponente entnommen werden muss.

Build Document [9, S. 12f] Nachdem eine Möglichkeit geschaffen wurde, die Rohdaten einer oder mehrerer Quellen einzulesen, muss definiert werden, wie diese Informationen zu Dokumenten zusammengefasst werden. Wie bereits erwähnt, können einzelne E-Mails als Dokumente definiert werden. In diesem Schritt werden die Dokumente erstellt und deren Felder mit Werten befüllt. Die Felder können frei definiert werden, bei E-Mails würden sich Absender, Empfänger, Betreff und die Nachricht als Feld anbieten. Diese Einteilung hängt jedoch davon ab, welche Daten für die Suchanfragen relevant sind. Dieser Schritt ist nicht Teil von Apache Lucene, sondern muss vom Applikationsentwickler implementiert werden.

Analyze Document [9, S. 13] Jedes Dokument wird in diesem Schritt in sogenannte „Tokens“ aufgeteilt, welche in etwa einzelnen Wörtern entsprechen. Dieser Schritt wird von Lucene durchgeführt, kann jedoch sehr genau gesteuert und den benötigten Eigenschaften angepasst werden. In diesem

Schritt können unter anderem eine Rechtschreibprüfung erfolgen oder Synonyme für bestimmte Wörter eingebaut werden. Als Beispiel werden hier die Wörter „laptop“ und „notebook“ genannt, die in vielen Fällen gleichbedeutend sind. In diesem Schritt ist es wichtig, genau die Erfordernisse der Suche zu kennen, um die Dokumente entsprechend dieser anzupassen.

Index Document [9, S. 14] Der letzte Schritt in diesem Bereich ist die eigentliche Indizierung der Dokumente. Hierbei werden die vorbereiteten Dokumente in den Suchindex eingetragen. Dies geschieht großteils automatisiert, kann jedoch wiederum an die Erfordernisse angepasst werden.

Suchanfrage Das Senden von Suchanfragen besteht aus vier Schritten, wie in Abbildung 2.4 dargestellt ist. Diese werden im Folgenden erläutert.

Search User Interface [9, S. 14f] Die Benutzeroberfläche dient sowohl zur Eingabe der Suchbefehle als auch zur Darstellung der Suchergebnisse. Bei der Gestaltung der Eingabemaske ist es wichtig, die Oberfläche simpel zu halten, detaillierte Suchoptionen sollten auf einer eigenen Benutzeroberfläche zusammengefasst werden, um den Benutzer nicht zu überfordern.

Die Darstellung der Suchergebnisse sollte klar strukturiert und mit wenig Text erfolgen. Wenn die Suchfunktion im Hintergrund Aktionen wie das Verwenden von Synonymen oder das Ausbessern von Tippfehlern durchführt, sollte dies dem Benutzer mitgeteilt werden.

Build Query [9, S. 15] In diesem Schritt werden die vom Benutzer eingegebenen Suchparameter in „Query“-Objekte umgewandelt. Lucene bietet dafür den sogenannten „QueryParser“, der dies automatisch durchführt. Die Funktionsweise des „QueryParser“ kann an die Erfordernisse einer Anwendung angepasst werden.

Run Query [9, S. 16] Das Ergebnis dieses Schrittes ist eine Sammlung von Suchergebnissen, die der übergebenen „Query“ entsprechen. Ebenso wie der „QueryParser“ kann diese Komponente an die geforderten Funktionen angepasst werden. Theoretisch betrachtet existieren drei verschiedene Modelle einer Suche:

- **Pure Boolean model:** Suchergebnisse bzw. Dokumente entsprechen entweder der Suchanfrage oder nicht. Dabei wird keine Relevanz der Suchergebnisse berechnet und nur die den Suchparametern entsprechenden Ergebnisse zurückgeliefert.
- **Vector Space model:** Die Suchparameter werden ebenso wie die indizierten Dokumente als Vektoren modelliert, wobei jeder Parameter einer Dimension entspricht. Bei einer Suchanfrage wird die Distanz

zwischen dem Suchvektor und den im Index enthaltenen Dokumentenvektoren berechnet.

- **Probabilistic model:** Bei diesem Modell wird die Wahrscheinlichkeit berechnet, ob die Suchanfrage auf ein im Index enthaltenes Dokument zutrifft.

Lucene verwendet eine Kombination aus einem „Pure Boolean model“ und einem „Vector Space model“ [9, S. 16].

Render Results [9, S. 16] Das Rendern der Resultate dient dazu, die Suchergebnisse in einer für den Benutzer ansprechenden Weise darzustellen. Dabei sollte darauf geachtet werden, dass die durchgeführte Suche leicht nachjustiert werden kann.

2.4.2 Zend Search Lucene

Eine der Komponenten des Zend-Frameworks ist „Zend Search Lucene“, eine in reinem PHP geschriebene und an Apache Lucene angelehnte Volltext-Such-Engine. „Zend-Search-Lucene“ verwendet einen zur Apache-Version kompatiblen Suchindex[4], wodurch es möglich ist, beide Systeme parallel zu betreiben.

2.5 Relevante Teile der HTML-Spezifikation

Die HTML-Spezifikation beschreibt den Aufbau von HTML-Seiten, sowie alle darin erlaubten Elemente. Weiters definiert die Spezifikation durch eine „Document Type Definition“ (DTD), wie Elemente ineinander verschachtelt werden können.

Dieser Abschnitt beschreibt neben Teilen der HTML4-Spezifikation auch Änderungen im derzeit in Entwicklung befindlichen HTML5-Standard.

2.5.1 Element-Gruppen

In der HTML4-DTD werden alle Elemente und deren Attribute einer Gruppe zugeordnet. Tabelle 2.5 zeigt alle Element-Gruppen der HTML4-Spezifikation sowie die Tags der enthaltenen Elemente. Die DTD definiert anhand dieser Gruppen unter anderem die erlaubte Verschachtelung der Elemente. Anhand der in Listing 2.3 gezeigten Definition des „A“-Elementes ist ersichtlich, dass ein Link nur Elemente der Inline-Gruppe enthalten darf. Als einzige Ausnahme werden „nested links“, also Links innerhalb von Links definiert (vgl. Tabelle 2.5).

[4]Die Kompatibilität hängt von den verwendeten Versionen des Zend-Frameworks sowie Apache Lucene ab. In [44] wird dies näher beschrieben.

Gruppe	**Beschreibung**	**Tags**
Phrase	Tags zur logischen Auszeichnung im Text. Der Browser entscheidet, wie diese Elemente dargestellt werden.	, , <sfn>, <code>, <samp>, <kbd>, <var>, <cite>, <abbr>, <acronym>
Fontstyle	Tags zur direkten Formatierung eines Textes	<tt>, <i>, <b>, <big>, <small>
Special	Spezial-Tags	<a>, <img>, <object>, , <script>, <map>, <q>, <sub>, <sup>, , <bdo>
Formctrl	Formular-Elemente	<input>, <select>, <textarea>, <label>, <button>
Heading	Überschriften	<h1> bis <h6>
List	Listen	,
Preformatted	Vorformatierter Inhalt	<pre>
Inline	Alle Elemente, die innerhalb eines anderen Elementes vorkommen können	Alle Tags der Phrase-, Fontstyle-, Special- und Formctrl-Gruppe mit Ausnahme des „A“-Elementes [38, Abschnitt 12.2.2].
Block	Block-Level Elemente	Die Elemente der Heading-, List- und Preformatted-Gruppe sowie die Elemente <p>, <blockquote>, <dl>, , <noscript>, <blockquote>, <form>, <hr>, <table>, <fieldset>, <address>

Tabelle 2.5: HTML4 Element-Gruppen [32, Abschnitt 21]

Als weiteres Beispiel ist in Listing 2.3 das „SPAN“-Element der Special-Gruppe angeführt. Wie daraus zu entnehmen ist, kann das „SPAN“-Element weitere Elemente der Inline-Gruppe enthalten. Die HTML4 Spezifikation definiert kein Maximum zur Verschachtelung von Elementen, praktische Versuche im Internet wie in [20] oder [36] deuten jedoch darauf hin, dass eine browserspezifische Grenze existiert. Praktisch hat diese Einschränkung jedoch keine Bedeutung, da sie auf gut strukturierten Internetseiten nie erreicht wird.

Listing 2.3: DTD-Auszug des „A“- und „SPAN“-Elementes [38]

```
1 <!ELEMENT A - - (%inline;)* -(A) -- anchor -->
2 <!ELEMENT SPAN - - (%inline;)* -- generic language/style container -->
```

2.5.2 HTML5

Dieser Abschnitt soll einen kurzen Überblick über die relevanten Veränderungen von HTML5 gegenüber der Vorgängerversion bieten.

Änderungen von Elementen

HTML5 definiert einige neue Elemente, die unter [26, Abschnitt 3.1] aufgelistet sind. Die für die vorliegende Arbeit relevanten Elemente sind in Tabelle 2.6 angeführt.

Element-Tag	Beschreibung
<article>	Markiert einen Nachrichten-Artikel oder Blog-Eintrag innerhalb einer Internetseite
<nav>	Zeichnet eine Navigations-Leiste aus
<track>	Definiert einen Beschreibungs-Text zu einem Video
<command>	Beschreibt einen Befehl, den der Benutzer anwenden kann

Tabelle 2.6: Relevante neue Elemente in HTML5 [26, Abschnitt 3.1]

Änderungen von Attributen

In HTML5 werden sowohl neue Attribute als auch neue Werte für einige Attribute definiert. Insbesondere das „input“-Element wurde durch neue Werte für das „type“-Attribut erweitert [26, Abschnitt 3.1]. Relevant ist dabei vor allem der Typ „search“, mit dessen Hilfe Eingabefelder für eine Suche definiert werden können. Alle weiteren Typen sind in [26, Abschnitte 3.1] aufgelistet.

Die sonstigen Änderungen an den Attributen sind in [26, Abschnitte 3.4 - 3.6] aufgelistet, für die vorliegende Arbeit jedoch nicht relevant.

Verschachtelungen von Elementen

Die HTML5-Spezifikation enthält einige Änderungen an den Verschachtelungsmöglichkeiten von Elementen. Eine der Änderungen ist die Einführung des „transparent content model“. Es definiert, dass Elemente die dieses „content model“ besitzen, alle Elemente beinhalten dürfen, die deren Eltern-Elemente beinhalten darf. Dies wird unter anderem für Links definiert [26,

Abschnitte 4]. Auch wurden die Element-Gruppen dahingehend überarbeitet, dass ein Element sowohl keiner, einer oder auch mehreren Gruppen angehören kann.

Kapitel 3

Related Work

Zum Thema der Sprachsteuerung von Internetseiten gibt es bereits Projekte, die ähnliche Ziele verfolgen. An dieser Stelle werden die Grundsätze dieser Projekte und deren Abgrenzung zum im Rahmen der Masterarbeit umgesetzten Masterprojektes beschrieben.

3.1 The WAMI Toolkit

Das „WAMI-Toolkit“[1]ist eine Plattform zur Entwicklung von multimodalen Web-Anwendungen. Multimodal bedeutet in diesem Zusammenhang, dass die Interaktion mit der Anwendung über verschiedene Ein- und Ausgabemethoden möglich ist.

3.1.1 Komponenten und Funktionsweise

Die WAMI-Plattform besteht aus einer Server- und einer Client-Komponente. Während die Client-Komponente zur Steuerung der Internetseite zuständig ist, kümmert sich die Serverseite um die Spracherkennung. Die einzelnen Komponenten werden nachfolgend erläutert.

Spracherkennung

Die Spracherkennung erfolgt serverseitig auf einem Java Applikations-Server, der nicht näher spezifiziert wird. Applikationen, die das „WAMI-Toolkit“ verwenden, müssen ein spezifisches Sprachmodell sowie die dazugehörige Sprachgrammatik im JSGF-Format bereitstellen [4, S. 2f].

Mikrofonzugriff

Der Mikrofonzugriff wurde in früheren Versionen mit einem Java-Applet realisiert, das am Rechner des Clients ausgeführt wird [4, S. 2f]. Dieses App-

[1]Die Abkürzung WAMI steht für „Web-Accessible Multimodal Applications“

let nimmt die Audio-Daten auf und sendet diese direkt an den Server, der die Spracherkennung durchführt. In der später erschienenen Version „WAMI 2.0“ wurde das Java-Applet mit einer Flash-basierten Lösung ersetzt [23, Abschnitt 'August 2011, Meet WAMI 2.0'].

3.1.2 Abgrenzung

Das „WAMI-Toolkit“ bietet eine Möglichkeit, bestehende Web-Anwendungen und Internetseiten mittels Sprachbefehlen steuerbar zu machen. Dabei muss sowohl die Steuerungslogik als auch die zu verwendende Sprachgrammatik vom Anwendungsentwickler bereitgestellt werden. Der Ansatz des Masterprojektes baut hingegen auf dem Grundsatz auf, dass eine Internetseite ohne zusätzliche Software wie Java oder Flash und ohne die Bereitstellung der Sprachgrammatik steuerbar sein soll. Die grundsätzliche Idee einer multimodalen Steuerung sowie die Ausführung der Steuerungsaktionen im Browser des Clients mittels JavaScript ist der Architektur des Masterprojektes ähnlich.

3.2 Online Speech Recognition API

Die so genannte „Online Speech Recognition API“ bietet ebenso wie das WAMI-Toolkit die Möglichkeit, eine Spracherkennungskomponente in bestehende Internetseiten einzufügen. Nachfolgend wird das Projekt näher beschrieben und dessen Funktionsweise erläutert.

3.2.1 Komponenten und Funktionsweise

Die „Online Speech Recognition API“ bietet neben einer Spracherkennung auch einen Sprachsynthesizer an, der serverseitig Text in Sprache umwandelt und an den Benutzer sendet. Nachdem die Spracherkennung erfolgt ist, kann mittels „Callback-Funktionen“ erkannt werden, ob sie erfolgreich war [35]. Wenn dies der Fall ist, kann in der entsprechenden Funktion eine Steuerung der Internetseite erfolgen.

Spracherkennung

Die Spracherkennung erfolgt serverseitig und benötigt eine Sprachgrammatik im JSGF Format. Der Server wird nach der Registrierung zur Verfügung gestellt und muss daher nicht selbst konfiguriert werden. Ein Sprachmodell ist bereits am Server hinterlegt und muss nicht bereitgestellt werden. Die Sprachdaten müssen in 8khz und 16Bit kodiert sein.

Mikrofonzugriff

Der Mikrofonzugriff erfolgt, wie beim WAMI-Toolkit mit einer Flash-Komponente, die per JavaScript initialisiert wird.

3.2.2 Abgrenzung

Ebenso wie das WAMI-Toolkit wird der Mikrofon-Zugriff über eine Flash-Komponente realisiert. Dies widerspricht dem Ansatz der vorliegenden Arbeit, das Projekt ohne Plug-Ins im Browser des Benutzers zu realisieren. Weiters muss die Sprachgrammatik manuell erstellt werden, was bei umfangreichen Internetseiten sehr aufwendig ist.

Kapitel 4

Generierung der Sprachbefehle

In diesem Kapitel wird erläutert, wie aus der Struktur von HTML-Seiten die benötigten Informationen zur Generierung von Sprachbefehlen extrahiert werden können. Dabei wird auf die Richtlinien der Barrierefreiheit aus Abschnitt 2.3 sowie auf die HTML „Document Type Definition“ (DTD) Bezug genommen. Wie in Abschnitt 5.5 später beschrieben wird, basiert die Interaktion mit einer Internetseite hauptsächlich auf der Verwendung einer Maus zum Verwenden von klickbaren Elementen. Klickbar sind hierbei vor allem Links, Buttons sowie alle Elemente, die mittels „onclick“-, „onmousedown“- oder „onmouseup“-Attribut JavaScript-Code definieren, der bei den entsprechenden Events ausgeführt wird.

4.1 Ansatz

Der grundlegende Ansatz zur Generierung der Sprachbefehle ist es, aus den einzelnen HTML-Elementen eine schriftliche Beschreibung zu extrahieren und diese mit Schlüsselwörtern zu ergänzen, die der natürlichen Sprache entsprechen. Dabei wird für ein einzelnes Element mindestens ein Sprachbefehl generiert. Diese Befehle werden anschließend mit den in Abschnitt 5.4.1 erwähnten Meta-Informationen in den Lucene-Suchindex geschrieben.

Der Ablauf der Generierung besteht aus drei Schritten:

1. Dem Auffinden der Elemente, die für die Interaktion mit der Internetseite relevant sind,
2. dem Analysieren der Elemente nach ein oder mehreren Textwerten, die das Element beschreiben und
3. dem Finden von Schlüsselwörtern, mit denen diese Textwerte erweitert werden.

Zusätzlich ist es notwendig, sowohl Änderungen der verwendeten Elemente sowie das Generieren und Entfernen von Elementen zu überwachen. Diese beiden Schritte werden in Abschnitt 5.6 beschrieben. Für jede dieser

Änderungen muss die Generierung der Sprachbefehle erneut durchgeführt werden.

4.2 Auslesen der Textwerte

Die grundlegende Aufgabe des Systems ist die Generierung einer schriftlichen Repräsentation der relevanten HTML-Elemente. Dazu werden im Folgenden die einzelnen Elemente und deren Varianten näher betrachtet.

4.2.1 Links

Links gehören zu den am häufigsten verwendeten HTML-Elementen zur Interaktion mit Internetseiten. Der klickbare Bereich eines Links kann aus reinem Text, Bildern sowie allen "inline"-Elementen bestehen, wobei verschachtelte Links nicht erlaubt sind (vgl. Abschnitt 2.5). Das Auslesen der entsprechenden Werte hängt wesentlich mit dem Inhalt der Links sowie dem Kontext ab, in dem sie stehen. Da der klickbare Bereich eines Links aus beinahe allen HTML-Elementen bestehen kann (vgl. Abschnitt 2.5), muss das Auslesen des Textes rekursiv erfolgen.

Text-Links

Text-Links sind die einfachste Variante von Links, da sie keine weiteren HTML-Elemente enthalten. Im Link-Text sind jegliche Sonderzeichen aller Zeichenkodierungen erlaubt, die jeweils verwendete Kodierung muss jedoch als Meta-Information in der Seite vorhanden sein. Link-Texte beschreiben im Normalfall die Seite oder Aktion, die bei der Benutzung des Links geöffnet bzw. durchgeführt wird. Aus diesem Grund kann der Link-Text grundsätzlich direkt als Teil eines Sprachkommandos verwendet werden.

Eine Ausnahme hierbei bilden Links, deren Link-Texte nur aus „Read more“ oder „Click here“ bestehen und es somit meist keine Möglichkeit gibt, sinnvolle Informationen zur Generierung der Sprachkommandos zu extrahieren. Zieht man jedoch Richtlinie 13 der WCAG1 in Betracht, sind derartige Link-Texte zu vermeiden (vgl. Abschnitt 2.3.2).

Solche Link-Texte werden häufig in Blogs verwendet, in denen lange Artikel in der Übersichtsseite nur gekürzt dargestellt und erst ein Klick auf einen derartigen Link zum vollständigen Artikel führt. Ein Beispiel hierfür zeigt Abbildung 4.1, die dazugehörige HTML-Struktur ist in Listing 4.1 dargestellt.

Wie anhand der HTML-Struktur ersichtlich ist, bietet der HTML-Link ausgenommen vom Link-Ziel keine Informationen darüber, wohin er führt und kann daher nur schwer als Teil eines Sprachbefehles verwendet werden.

Listing 4.2 zeigt die HTML-Struktur des Titel-Links aus Abbildung 4.1. Wie aus den beiden Listings ersichtlich ist, führen beide Links zum gleichen

Ziel. In diesem Fall gibt es zwei Möglichkeiten zur weiteren Vorgehensweise:

1. ignorieren des „Read more“-Links
2. oder generieren eines eigenen Sprachbefehls durch die Kombination des Titel-Links und des Inhaltes des „Read more“-Links.

Letzteres erfordert das Erkennen der Zusammengehörigkeit dieser beiden Links, was durch den Vergleich der Link-Ziele möglich ist. Anschließend wird aus beiden Links die jeweilige textliche Repräsentation generiert und ein oder mehrere kombinierte Sprachbefehle erstellt.

Abbildung 4.1: Beispiel für einen Blog-Artikel mit „Read more“-Link

Listing 4.1: HTML-Code des in Abbildung 4.1 dargestellten Links

```
1 <a href="http://www.engadget.com/2012/05/03/marketplace-for-windows-22-
      new-countries/">
2   Read the full post on
3   <span>mobile.engadget.com</span>
4 </a>
```

Listing 4.2: HTML-Code des in Abbildung 4.1 dargestellten Titel-Links

```
1 <a href="http://www.engadget.com/2012/05/03/marketplace-for-windows-22-
      new-countries/">
2   Web Marketplace for Windows Phone gets 22 more stamps in its passport
3 </a>
```

Bilder-Links

Links, deren klickbarer Teil entweder vollständig oder teilweise aus einem oder mehreren Bildern besteht, können anders als reine Text-Links nicht direkt verwendet werden. Um eine schriftliche Repräsentation aus einem Link zu generieren, kann der Wert des „alt“-Attributes verwendet werden, der nach WCAG1 einer schriftlichen Beschreibung des Bildes entsprechen soll (vgl. Abschnitt 2.3.2). Ein Beispiel für einen entsprechenden Wert für das

Abbildung 4.2: Einige Euro-Münzen

in Abbildung 4.2 dargestellte Bild wäre „a few euro coins“ oder „a picture of some euro coins“.

Bei Bildern, die keine schriftliche Beschreibung besitzen, ist das Auslesen des Dateinamens eine weitere Möglichkeit, Informationen über das Bild herauszufinden. Auf Internetseiten, die für Suchmaschinen optimiert sind, lässt sich aus dem Dateinamen der Inhalt eines Bildes ableiten. Dazu wird unter anderem in einem Blog-Eintrag in [25] oder in einer Antwort zu einer entsprechenden Frage auf Stackoverflow in [14] geraten. Ein möglicher Dateiname des Bildes in Abbildung 4.2 wäre „euro_coins.png“, woraus nach dem Entfernen aller Unterstriche oder anderer Sonderzeichen die Stichwörter „euro“ und „coins“ abgeleitet werden können, welche den Kontext des Bildes beschreiben.

Links bestehend aus sonstigen Elementen

Die HTML-Spezifikation erlaubt es, beinahe alle HTML-Elemente als klickbaren Bereich eines Links zu definieren. Bei Elementen, die eine Formatierung oder Auszeichnung eines Textes durchführen, ist dies nicht problematisch. Da jedoch auch alle Elemente der „Formctrl“-Gruppe erlaubt sind, können auch Textfelder oder Radiobuttons für Links verwendet werden. Da dies der Erwartung des Benutzers widerspricht, werden derartige Konstrukte praktisch nicht verwendet. Die meisten der in Tabelle 2.5 gelisteten Elemente können zur Auszeichnung von Texten verwendet werden, lediglich die Elemente der „Formctrl“-Gruppe sind speziell zu behandeln. Eine Ausnahme dieser Gruppe ist das Label-Element, das im nachfolgenden Abschnitt näher betrachtet wird. Formular-Elemente werden in Abschnitt 4.2.3 behandelt.

Das Label-Element

Das Label-Element dient zur schriftlichen Beschreibung eines Elementes, welches über eine ID im „for“-Attribut des Labels referenziert wird. Der Text des Labels kann als zusätzlicher Teil eines Sprachbefehles jedes relevanten Elementes verwendet werden. Da die in der HTML-Spezifikation vorgesehene Referenzierung durch das „for“-Attribut einseitig ist, definiert „WAI-ARIA“ das „aria-labelledby“-Attribut, das vom beschriebenen Element aus das Label über eine ID referenziert (vgl. Abschnitt 2.3.4).

4.2.2 Buttons

Buttons sind Elemente, die über Mausklicks verwendet werden, um bestimmte Funktionen aufzurufen. Dabei unterscheidet man Submit- und Reset-Buttons sowie all jenen, die JavaScript-Code ausführen. Submit- und Reset-Buttons dienen dem Abschicken bzw. Zurücksetzen von Formularen und werden daher nur innerhalb dieser verwendet. Die Verwendung von Formularen wird in Abschnitt 4.2.3 beschrieben.

Die Berechnung einer schriftlichen Repräsentation eines Buttons ist meist auf die Beschriftung, den Titel und den Wert des „alt“-Attributes beschränkt.

4.2.3 Formulare

Um eine schriftliche Beschreibung eines Formulars zu berechnen ist es nötig, dessen Zweck zu kennen. Wenn ein Formular beispielsweise zum Absenden eines Gästebucheintrags dient, können die später verwendeten Schlüsselwörter entsprechend angepasst werden. Da die HTML4-Spezifikation das „alt“-Attribut nicht das Formular-Element definiert, kann zum Bestimmen des Kontextes nur der Name und der Titel des Formulares herangezogen werden. Da die Namen von Formularen jedoch zur Identifizierung innerhalb von JavaScript-Code verwendet werden, ist es nicht üblich dabei Leer- oder Sonderzeichen zu verwenden.

Der Titel hingegen kann eine genauere Beschreibung des Formulars enthalten, die für Sprachbefehle verwendet werden kann. Bezogen auf das Beispiel eines Gästebuches, könnte der Titel des Formulars „Guestbook Form“ sein. Aus diesem Kontext heraus können die Eingabe-Elemente des Formulars identifiziert werden.

Input-Elemente

Input-Elemente dienen zur Eingabe von Daten in Formularen. Über das „type“-Attribut kann die Art des Eingabefeldes definiert werden. Das Berechnen der schriftlichen Repräsentation ist für alle Input-Elemente ähnlich. Da die Elemente einen Namen, einen Typ und eine alternative Beschreibung

besitzen können, werden diese Attribute bei Vorhandensein zur Beschreibung herangezogen.

Seit HTML5 können Input-Elemente neben den bereits in der Version 4 definierten noch weitere Typen annehmen (vgl. Abschnitt 2.5.2). Insbesondere die Typen „email“ oder „search“ können zum Bestimmen des Kontextes eines Eingabefeldes dienen.

Radio-Buttons und Select-Boxen

Anders als bei den restlichen Eingabefeldern haben Radio-Buttons und Select-Boxen eine festgelegte Auswahl an Optionen. Daher muss sowohl für das Element selbst als auch für alle Optionen eine Beschreibung extrahiert werden.

4.2.4 Benutzerdefinierte Widgets

Im Gegensatz zu den im HTML-Standard definierten Elementen werden insbesondere in modernen „Rich Internet Applications“ selbst geschriebene Widgets verwendet. Diese werde im Verhalten an die nativen Elemente wie z.B. Select-Boxen angepasst, wodurch der Benutzer in der Regel keinen Unterschied bemerkt. Ein automatisiertes Auslesen der Funktion dieser Elemente ist nicht ohne Weiteres möglich.

Neben der Möglichkeit, CSS-Klassen zur Klassifizierung von Widgets heranzuziehen, können vor allem das „WAI-ARIA“ Rollenkonzept dazu verwendet werden, die Eigenschaften von Widgets zu bestimmen (vgl. Abschnitt 2.3.4).

4.3 Nachbearbeitung der Werte

Nachdem aus den einzelnen Elementen mindestens ein Textwert extrahiert wurde, ist es nötig einige Teile dieser Texte zu ersetzen, um einen sinnvollen Vergleich des vom Spracherkennungssystem gelieferten Wertes zu ermöglichen. Dies hat vor allem die folgenden technischen Gründe:

1. Die Spracherkennung von Google liefert in den meisten Fällen keine Sonderzeichen zurück. Eine Ausnahme ist hierbei beispielsweise die Kurzform „it’s“ für „it is“, bei der das Anführungszeichen korrekt erkannt wird. Aus diesem Grund werden Sonderzeichen vor der weiteren Verwendung durch die entsprechenden englischen Wörter ersetzt (vgl. Abschnitt 4.3.1 und 4.3.2).
2. Die Spracherkennung liefert Zahlen, insbesondere jene mit Kommastellen nicht konsistent zurück. Dies wird in Abschnitt 4.3.3 näher beschrieben.

4.3.1 Ersetzen von Sonderzeichen

Insbesondere durch die Spracherkennung, die in den erkannten Texten beinahe keine Sonderzeichen zurückliefert, müssen Sonderzeichen durch einzelne Zeichen oder in manchen Fällen durch entsprechende Worte ersetzt werden. Tabelle 4.1 enthält einige Sonderformen von Vokalen, die durch die jeweilige Grundform ersetzt werden müssen. Benötigt wird dies, weil das englische Alphabet keine Umlaute und Sonderzeichen enthält.

Zeichen	Ersatz-Zeichen
á, â, ă, å, ...	a
é, ê, ĕ, ė, ...	e
ó, ô, ö, ø, ...	o

Tabelle 4.1: Umlaute und deren Ersatzzeichen

Da die Google-Spracherkennung gesprochene Wörter wie „euro“ oder „dollar“ nicht als die Sonderzeichen „€“ bzw. „$“ erkennt, muss in den Sprachbefehlen die Textform verwendet werden. Tabelle 4.2 listet einige dieser Fälle auf.

Zeichen	Ersatz-Wort(e)
€	„euro“ bzw. „euros“
$	„dollar“ bzw. „dollars“
@	„at“
%	„percent“
&	„and“
+[1]	„plus“
#	„sharp“

Tabelle 4.2: Sonderzeichen und deren Ersatzworte

4.3.2 Entfernen von Anführungszeichen

Anführungszeichen am Anfang und Ende eines Wortes oder Satzes werden von der Spracherkennung nicht erkannt. Aus diesem Grund müssen alle Anführungszeichen in Link-Texten entfernt werden. Die einzige Ausnahme hierfür sind Anführungszeichen inmitten eines Wortes, wie beispielsweise beim Wort „it’s“.

[1] Diese Ersetzung ist nicht immer notwendig. In einem mathematischen Kontext wird beispielsweise die gesprochene Aussage „five plus five“ korrekt als „5 + 5“ erkannt, wohingegen bei der Aussage „myself plus you“ das Wort „plus“ nicht durch das „+“-Zeichen ersetzt wird. Aus diesem Grund muss die Ersetzung nur durchgeführt werden, wenn es sich nicht um einen mathematischen Ausdruck handelt.

4.3.3 Zahlen

Die Erkennung von Zahlen ist eine der Grundfunktionalitäten jeder Spracherkennung und ist daher meist sehr ausgereift. Da jedoch Zahlen nicht immer gleich dargestellt werden, ist es nötig, die für ein Sprachkommando relevanten Zahlen entsprechend zu formatieren. Dies betrifft neben Zahlengruppierungen und Kommazahlen auch rationale Zahlen, die in Bruchform vorkommen.

Zahlengruppierung

Um die Lesbarkeit von langen Zahlen zu erhöhen, werden diese auf verschiedene Arten gruppiert. Häufig verwendet wird die Gruppierung mit einem „Tausender-Trennzeichen“, das von rechts ausgehend nach je drei Ziffern ein bestimmtes Zeichen einfügt. Dazu werden Punkte, Leerzeichen oder Beistriche verwendet. Die Zahl **42500150** kann je nach Trennzeichen als **42.500.150**, **42,500,150** oder **42 500 150** dargestellt werden.

Die jeweilige Darstellungsform, die auf einer Internetseite verwendet wird, muss in das vom Spracherkennungssystem zurückgelieferte Format umgewandelt werden, um eine korrekte Erkennung der Zahl zu ermöglichen. Die im in Kapitel 5 verwendete Google-Spracherkennung liefert Zahlen mit Beistrichen als Tausender-Trennzeichen sowie einem Punkt als Dezimaltrennzeichen. Tabelle 4.3 zeigt einige gesprochene Zahlen sowie den jeweils von der Spracherkennung zurückgelieferten Wert.

Wie aus der Tabelle ersichtlich ist, werden Kommazahlen nicht immer korrekt an die Zahl angehängt, sondern als eigenständiger Wert betrachtet. Dabei wird die Kommazahl immer dann korrekt an die Zahl angehängt, wenn darin kein Tausender-Trennzeichen enthalten ist. Diese Aussage kann jedoch mangels genauer Dokumentation der Funktionsweise der Google-Spracherkennung nicht belegt werden. Durch empirische Versuche konnten keine Ausnahmen dieser Regel gefunden werden.

Gesprochene Aussage	**Erkannter Wert**
„five million“	5,000,000
„one point five“	1.5
„five million point five“	5,000,000 .5
„one thousand point five“	1000.5
„ten thousand point five“	10,000 .5

Tabelle 4.3: Beispiele für Zahlen der Google Spracherkennung

Bruchzahl	Ersatz-Wort(e)
$\frac{1}{2} bzw. 1/2$	„(one) half“
$\frac{1}{4} bzw. 1/4$	„(one) quarter“ bzw. „one out of four“
$\frac{3}{8} bzw. 3/8$	„three-eighths“

Tabelle 4.4: Bruchzahlen und deren Ersatz-Worte

Rationale Zahlen

Für rationale Zahlen ist im Falle einer Darstellung als Bruch eine Ersetzung notwendig. Tabelle 4.4 zeigt dazu einige Beispiele.

4.4 Schlüsselwörter

Wie bereits erwähnt werden die schriftlichen Repräsentationen der relevanten HTML-Elemente mit Stichwörtern angereichert, um die Treffergenauigkeit bei der Suche mit Lucene zu erhöhen. Welche Schlüsselwörter verwendet werden, hängt von mehreren Einflüssen ab, die in diesem Abschnitt beschrieben werden.

4.4.1 Standard-Schlüsselwörter

Als Standard-Schlüsselwörter sind Wörter gemeint, die bei nahezu jedem Sprachbefehl Verwendung finden. Die verwendeten Wörter hängen unter anderem davon ab, welchen Zweck ein bestimmtes Element hat.

Links mit „href“

Links, die ein Link-Ziel mit dem „href“-Attribut definieren, haben den Zweck, eine bestimmte Internetseite aufzurufen. Eine Ausnahme hiervon sind Links, deren Wert des „href“-Attributes aus einem Raute-Zeichen („#“) besteht. Da es sich bei allen anderen Links um einen Navigationsvorgang handelt, werden dabei entsprechende Stichwörter verwendet. Dazu zählen die folgenden Wörter:

- „go to“
- „navigate to“
- „open“
- „show me“
- „click on“

Die schriftliche Repräsentation eines Links oder Buttons wird mit jedem dieser Stichwörter oder mit Stichwortpaaren kombiniert. Daraus ergeben sich

mindestens fünf Sprachbefehle für jedes Element. Für Links einer Navigationsleiste sind vor allem die ersten zwei bzw. drei Stichwort-Kombinationen dieser Liste relevant.

Links mit „onclick“

Links, die auf ein „click“-Ereignis reagieren und JavaScript-Code ausführen, dienen meist keinen Navigationszwecken, sondern führen beispielsweise Operationen am DOM-Baum aus oder senden AJAX-Requests an einen Server. Stichwörter wie „go to“ oder „navigate to“ sind in diesem Fall nicht passend. Da es sich bei diesen Links um „Aktions-Links“ handelt, werden die Stichwörter angepasst:

- „click on“
- „use“
- „trigger“

Für den Fall, dass ein Link oder ein Button ein Formular absendet, werden die folgenden Stichwörter zusätzlich verwendet:

- „submit form“
- „send form“

4.4.2 Einflüsse auf die Schlüsselwörter

Die Verwendung weiterer Schlüsselwörter hängt von einigen Faktoren ab, die nachfolgend erläutert werden.

Thema der Internetseite

Das Thema der Internetseite beeinflusst die Schlüsselwörter, in dem es beschreibt, **was** ein Link öffnet. Nimmt man die Internetseite der BBC als Beispiel, führen beinahe alle Links zu Nachrichten-Artikeln (engl. „news articles“ oder nur „articles“).

Anhand des Links „Twist in 'Bin laden doctor' case“ könnte die Entwicklung eines Sprachbefehles in den Folgenden Schritten erfolgen:

1. Auslesen des gesamten Textes „Twist in 'Bin laden doctor' case“
2. Entfernen der Anführungszeichen (vgl. Abschnitt 4.3.2)
3. Erkennung, ob es sich bei der Seite um eine Nachrichten-Seite handelt
4. Hinzufügen der Standard- und der spezifischen Stichwörter

Aus dieser Abfolge entstehen mehrere Sprachbefehle. Zwei Beispiele dafür sind:

- „go to the twist in bin laden doctor case article“
- „open the bin laden doctor twist news article“.

Ansätze, wie das Thema einer Internetseite bestimmt werden kann, werden in Abschnitt 4.7 beschrieben.

Ziel des Links

Um das Ziel eines Links zu identifizieren, kann sowohl der Kontext als auch der Wert des „href“-Attributes herangezogen werden. Dabei können wiederum die in Abschnitt 4.7 beschriebenen Methoden verwendet werden, um das Ziel anhand der URL zu klassifizieren.

Zusätzlich kann erfasst werden, ob ein Ziel am gleichen Server wie die aktuelle Seite liegt. Da dies jedoch meist für den Benutzer ersichtlich ist, kann nicht davon ausgegangen werden, dass sich die Schlüsselwörter dabei zu seiteninternen Links unterscheiden.

Platzierung des Elementes

Internetseiten sind sehr häufig in zwei oder mehrere Bereiche strukturiert. Einem Kopfbereich, der meist das Logo der Internetseite und die Seiten-Navigation enthält, sowie dem Hauptbereich, der den Seiteninhalt beinhaltet. Dazu kommen oft Seitenleisten, die weitere Navigations-Links oder wie bei Blogs üblich eine „Blogroll“[2] und Kategorie-Links enthalten.

Für den Fall, dass keine Erkennung der Navigationsleiste z.B. durch WAI-ARIA Rollen möglich ist, könnte die Positionierung von Links im Kopfbereich der Seite ein Indiz darauf sein. Ebenso wäre die Positionierung eines Links im Hauptteil der Seite die Möglichkeit, Links zu Artikeln etc. zu erkennen, sofern das Thema bzw. die Art der Internetseite bekannt ist (vgl. Abschnitt 'Thema der Internetseite').

4.5 Formularelemente

Zur Verwendung von Formularelementen muss wie bereits in Abschnitt 4.2.3 zwischen Elementen mit freier Eingabe und jenen mit festgelegten Auswahlmöglichkeiten unterschieden werden.

Um ein Formularelement genauer zu bestimmen, ist es nötig, die Beschreibung des Formulars in die Sprachbefehle zu integrieren.

4.5.1 Textfelder

Bei der Verwendung von Textfeldern können zwei Vorgehensweisen angewandt werden:

- das Textfeld wird ausgewählt und anschließend über die Google-Spracherkennung mit einem Wert befüllt

[2] Eine Blogroll ist eine Sammlung von Links zu anderen Blogs.

- der Sprachbefehl enthält sowohl die Identifikation des Textfeldes als auch der Text, der in das Feld eingefügt werden soll.

Bezogen auf das bereits in Abschnitt 4.2.3 verwendete Beispiel eines Gästebuchformulars könnten zur Eingabe der E-Mail Adresse die folgenden Sprachbefehle definiert werden:

- „enter the email field inside the guestbook form“, um das Element zu fokussieren oder
- „enter john@doe.com into the email field inside the guestbook form“, um die E-Mail Adress direkt in das Feld einzufügen.

Aus Benutzersicht wäre die Kombination des Textfeldes sowie des einzufügenden Wertes in einem Sprachbefehl von Vorteil. Aufgrund der Tatsache, dass dieser Ansatz jedoch sehr viel Rechenaufwand zur Unterscheidung des Befehles sowie des Wertes benötigt, wurde er im Masterprojekt nicht umgesetzt.

4.5.2 Checkboxen

Da Checkboxen nur zwei Werte annehmen können (ausgewählt oder nicht ausgewählt), muss für diese Auswahlmöglichkeiten keine eigene schriftliche Beschreibung berechnet werden. Zur Interaktion mit einer Checkbox sind drei Aktionen erforderlich:

- das Auswählen bzw. Aktivieren
- das Abwählen bzw. Deaktivieren
- das Negieren des Wertes (engl. „toggle“).

Der aktivierte Zustand einer Checkbox wird mit „checked“ beschrieben, der deaktivierte Zustand mit „unchecked“. Aus diesen beiden Zuständen können nun die entsprechenden Verben für die Sprachbefehle hergeleitet werden. Basierend auf dem Gästebuchbeispiel, das eine Checkbox zum Abonnieren eines Newsletters enthält, können die folgenden Befehle verwendet werden:

- „check the subscribe newsletter checkbox inside the guestbook form“ um die Checkbox zu aktivieren
- „uncheck the newsletter checkbox within the guestbook form“ um die Checkbox zu deaktivieren oder
- „toggle the newsletter checkbox“ um den Wert zu negieren.

4.5.3 Select-Boxen und Radiobuttons

Die Sprachbefehle zur Auswahl einzelner Elemente bei Select-Boxen und Radiobuttons sind grundsätzlich gleich. Für Select-Boxen sind zusätzliche Befehle nötig, um die Auswahlliste zu öffnen und zu schließen. Da die Auswahlmöglichkeiten bei Radiobuttons immer sichtbar sind, ist dieser Befehl

Listing 4.3: Beispiel einer Selectbox und Radiobuttons zur Auswahl eines Landes

```
1 <!-- Select-Box -->
2 <select name="country" title="Select a country" >
3   <option value="at">Austria</option>
4   <option value="de">Germany</option>
5   <option value="ch">Switzerland</option>
6 </select>
7
8 <!-- Radiobuttons -->
9 <input type="radio" name="country" value="at" />Austria
10 <input type="radio" name="country" value="de" />Germany
11 <input type="radio" name="country" value="ch" />Switzerland
```

nicht notwendig. Basierend auf dem Beispiel aus Listing 4.3 können für die Steuerung der Select-Box die folgenden Befehle definiert werden:

- „open the country select box“, um die Auswahlliste zu öffnen,
- „close the country select box“, um sie zu schließen
- „select austria at the country select box“, um den Wert „Austria“ auszuwählen.

Letzterer Befehl kann auch für Radiobuttons verwendet werden, die Bezeichnungen des Formularfeldes müssen jedoch angepasst bzw. entfernt werden:

- „select the country austria“

4.6 Zusätzliche Befehle

Neben den Sprachbefehlen, deren Hauptteil aus der HTML-Seite ausgelesen werden kann, werden einige zusätzliche Befehle zur Steuerung des Browsers angeboten. Diese umfassen:

- die Vorwärts- und Rückwärtsnavigation
- das Neu-Laden der Seite
- das Schließen des Fensters
- das Aufrufen der „Home“-Seite des Browsers
- das Ausdrucken der Seite.

Da diese Funktionen mittels JavaScript ausgeführt werden können, ist es möglich, diese im Client-Controller zu implementieren. Die Sprachbefehle werden zur Unterscheidung von den restlichen Befehlen markiert, um die vom Sprach-Controller kommenden Steuerungsinformationen entsprechend anzupassen.

Die Definition der Sprachgrammatik, welche für oben genannte Aktionen in Listing 4.4 dargestellt ist, erfolgt in der JSGF-Form (vgl. Abschnitt 2.1.5).

Listing 4.4: JSGF-Grammatik der Sprachbefehle zur Browser-Interaktion

```
#JSGF V1.0;
grammar browserInteraction;

// common phrases
<politeness> = please;
<navigation> = go | navigate;
<page> = page | site;
<this> = this | current | active;

// back- and forward-navigation
<back-action> = [ <politeness> ] <navigation> back [ one | a ] <page> [ <politeness> ];
<forward-action> = [ <politeness> ] <navigation> forward [ one | a ] <page> [ <politeness> ];

// reloading the page
<refresh-phrases> = reload | refresh;
<refresh-action> = [ <politeness> ] <refresh-phrases> [ the | <this> | ( the | (active | current) ) ] [ <page> ] [ <politeness> ];

// print the page
<print-out> = print out | printout | print-out;
<print-out-action> = [ <politeness> ] make a <print-out> [ of ] [ <this> ] [ <page> ] [ <politeness> ];
<print-action> = [ <politeness> ] print [ <this> ] [ <page> ];

// go to home page
<home-action> = [ <politeness> ] <navigation> [ to the ] home <page> [ <politeness> ];

// close the window
<close-action> = [ <politeness> ] close [ the | <this> | (the | (active | current) ) ] [ browser ] ( window | tab ) [ <politeness> ];
```

Die im Listing als „common phrases“ gekennzeichneten Grammatikregeln dienen zur Vereinfachung der eigentlichen Sprachbefehle, in dem sie für ein Wort mehrere Alternativen bereitstellen.

4.6.1 Vor- und Zurück-Navigation

Zur Navigation zwischen HTML-Seiten können neben Klicks auf Links und Buttons auch die vom Browser angebotenen Vorwärts- und Rückwärtsschaltflächen verwendet werden. Diese navigieren durch eine vom Browser verwaltete „History“ in zwei Richtungen.

Zur Verwendung der beiden Navigationsschaltflächen werden zwei Regeln definiert, die einige optionale Teile enthalten. Da sich die Vorwärts- und Rückwärtsnavigation nur durch das Wort „back“ bzw. „forward“ unterschei-

den, können die folgenden Beispiele durch das Austauschen dieses Schlüsselwortes für die jeweils andere Navigationsrichtung verwendet werden.

Die folgenden Beispiele können aus der in Listing 4.4 definierten Grammatik abgeleitet werden:

- „go back“
- „navigate back please“
- „please navigate back one page“
- „navigate forward a site“

4.6.2 Neu-Laden der Seite

Zum Aktualisieren der aktuell angezeigten Seite bietet jeder Browser eine entsprechende Schaltfläche. Dabei wird eine erneute Anfrage an den Server für die Seite gesendet und diese anschließend angezeigt. Im Zuge dessen werden alle clientseitigen Änderungen, die mittels JavaScript durchgeführt wurden, zurückgesetzt, sofern diese nicht serverseitig oder in einem Cookie gespeichert wurden.

Da in der englischen Sprache im Bezug auf das Internet die Wörter „reload“ und „refresh“ meist gleichbedeutend verwendet werden, wird in der Grammatik in Listing 4.4 eine Regel namens „refresh-phrases“ definiert, die diese beiden Wörter als synonym zueinander setzt.

Einige Beispiele für Sprachbefehle die aus dieser Regel abgeleitet werden können, sind:

- „reload the current page“
- „please refresh this site“
- „reload current page please“

4.6.3 Schließen des Fensters

Die genaue Funktionsweise der JavaScript-Funktion zum Schließen von Fenstern ist abhängig vom verwendeten Browser. Wenn eine Seite innerhalb eines Tabs geöffnet wurde und der zu schließende Tab nicht der letzte eines Browser-Fensters ist, wird nur der Tab geschlossen, ansonsten das gesamte Browser-Fenster. Bei Browsern, die keine Tabs unterstützen, wie beispielsweise der Internet Explorer 6, wird immer das komplette Browser-Fenster geschlossen.

Je nach Browser sowie den individuellen Einstellungen ist das programmatische Schließen von Fenstern nicht erlaubt, wodurch diese Funktionalität für bestimmte Benutzer nicht verfügbar ist.

Aus der Grammatikregel „close-action“ aus Listing 4.4 lassen sich unter anderem die folgenden Sprachbefehle ableiten:

- „close window“
- „please close the active tab“

- „close current browser window please“

4.6.4 Aufrufen der „Home“-Seite

Zum Aufrufen der „Home“-Seite des Browsers werden einfache Navigationsbefehle verwendet. Aus der Regel „home-action“ aus Listing 4.4 können beispielsweise diese Befehle abgeleitet werden:

- „navigate to the home page please“
- „please go to the home page“

4.6.5 Ausdrucken der Seite

Um eine Internetseite auszudrucken, greift der Browser auf den systemweiten Druckdialog zurück, der die weitere Bearbeitung des Druckauftrages übernimmt. Zum programmatischen Aufruf des Druckdialogs kann auf eine JavaScript-Funktion zurückgegriffen werden, die gleichbedeutend mit einem Aufruf über das Programmmenü oder der Tastenkombination „Strg + P“ ist.

Die Sprachbefehle zum Drucken, die aus den Grammatikregeln „print-out-action“ und „print-action“ aus Listing 4.4 abgeleitet werden können, beinhalten unter anderem die folgenden:

- „make a print-out of this site“
- „print this page“

4.7 Klassifizierung einer Internetseite

Bei der Klassifizierung wird eine Internetseite anhand verschiedener Merkmale einer Kategorie zugewiesen. Da dieses Thema sehr umfangreich ist, werden verschiedene Ansätze nur kurz erwähnt und auf die entsprechenden wissenschaftlichen Arbeiten verwiesen.

In [7] wird eine Methode beschrieben, wie Internetseiten anhand der URL klassifiziert werden können. Dazu wird die URL in Segmente unterteilt und mit Methoden des maschinellen Lernens weiterverarbeitet. Glover et.al. beschreiben in [3] eine Klassifizierungsmethode, die mit „virtual documents“ arbeitet. Ein „virtual document“ ist dabei eine *„collection of anchortexts or extended anchortexts from links pointing to the target document“* [3]. Diese Methode ist ähnlich der von Google eingesetzten „Page-Rank“-Methode, mit dem Unterschied, dass nicht die Anzahl der Links auf eine Seite gezählt werden, sondern die jeweiligen Link-Texte analysiert werden.

4.8 Beispiel BBC

In diesem Abschnitt soll die Generierung von Sprachbefehlen am Beispiel der englischen Nachrichtenseite der BBC[3] gezeigt werden. Die Startseite, die in Abbildung 4.3 dargestellt ist, bietet neben der Hauptnavigation im oberen Bereich auch direkte Links zu Schlagzeilen und weiteren Artikeln (vgl. Abschnitt 4.8.1). Dazu werden sowohl Fotos als auch reine Text-Links verwendet. Neben statischen Links wird ein sogenanntes „Carousel“ verwendet, um ein Durchblättern von acht ausgewählten Schlagzeilen („Top News story“) zu ermöglichen (vgl. Abschnitt 4.8.2). Zusätzlich enthält die Seite sowohl eine Suche als auch eine Tabliste (vgl. Abschnitt 4.8.5 bzw. 4.8.4). Alle restlichen Links werden in Abschnitt 4.8.3 behandelt.

4.8.1 Navigation

Die Hauptnavigation, die in Abbildung 4.3 orange gekennzeichnet ist, besteht aus reinen Text-Links. Aus diesem Grund kann der Link-Text direkt ausgelesen und verwendet werden. Wie in Listing 4.5 erkenntlich, besitzt der „div“-Container der Navigationsleiste das „WAI-ARIA“-Attribut „role“ mit dem korrekten Wert „navigation“.

Ein Sonderfall dieser Navigationsleiste ist der Link mit dem Text „More“, der keine Navigation zu einer anderen Seite auslöst, sondern ein Menü mit weiteren Links öffnet. Dass es sich um keinen Navigations-Link handelt, kann programmatisch durch das Rautezeichen am Beginn des Link-Zieles erkannt werden. Es wird meist zur Kennzeichnung von Verweisen innerhalb einer Seite verwendet, wobei der Text hinter dem Rautezeichen dem Wert des „name“-Attributes eines Links bzw. „anchors“ entspricht [38].

Aufgrund der festgelegten Rolle des „div“-Elementes können nun die Stichwörter verwendet werden, die für eine Navigationsleiste vorgesehen sind (vgl. 4.4.1). Darauf aufbauend kann die Grammatik aus Listing 4.6 definiert werden. Die einzelnen Navigationslinks sind daher z.B. durch die folgenden Sprachbefehle ansteuerbar:

- „please go to sport“
- „open weather page please“
- „please show me the tv page“

Letzterer Befehl enthält den Artikel „the“, der nicht in der Grammatik definiert ist. Da sich durch diesen Artikel die Ähnlichkeit zwischen dem gesprochenen und dem in der Grammatik definierten Befehl nicht signifikant verändert, erzielt der Befehl durch die tolerante Lucene-Suche trotzdem eine sehr hohe Relevanz.

[3] http://www.bbc.com bzw. http://www.bbc.co.uk

Listing 4.5: Auszug aus dem HTML-Code der Navigation von BBC.com

```
<div id="'blq-nav"' role="'navigation"'>
  <ul id="'blw-nav-main"'>
    <li id="'blq-nav-news"'>
      <a href="'http://www.bbc.co.uk/news/"'>News</a>
    </li>
    ...
    <!-- Gekürzt -->
    ...
    <li id="'blq-nav-future"'>
      <a href="'http://www.bbc.co.uk/future/"'>Future</a>
    </li>
    <li id="'blq-nav-tv"'>
      <a href="'/tv/"'>TV</a>
    </li>
    <li id="'blq-nav-radio"'>
      <a href="'/radio/"'>Radio</a>
    </li>
    <li id="'blq-nav-more"' tabindex="'-1"'>
      <a href="'#blq-nav-more-links"' class="'istats-notrack"'>More</a>
      <span class="'blq-dropdown-arrow"'>
        <span></span>
      </span>
    </li>
  </ul>
</div>
```

Listing 4.6: Grammatik der BBC-Navigationsleiste

```
#JSGF V1.0
grammar navigation;

<politeness> = please;

<page> = page | site;
<keyword> = go to | open | show me;
<link-text> = news | sport | weather | travel | future | tv | radio;
<navigation> = [ <politeness> ] <keyword>  <link-text> [ <page> ] [ <
    politeness> ];
```

4.8.2 Top News Stories

Die „Top News Stories“ sind der in Abbildung 4.3 blau markierte Bereich, der je eine von acht Schlagzeilen inklusive eines Bildes darstellt. Mit Hilfe der beiden als Pfeil dargestellten klickbaren Flächen am linken und rechten Rand des Bereiches kann durch diese Schlagzeilen navigiert werden. Listing 4.7 zeigt die HTML-Struktur des nach links navigierenden Bereiches. Wie man erkennen kann, bestehen diese klickbaren Flächen weder aus Links oder Buttons noch aus Elementen mit „onclick“-Attributen. Daraus lässt sich

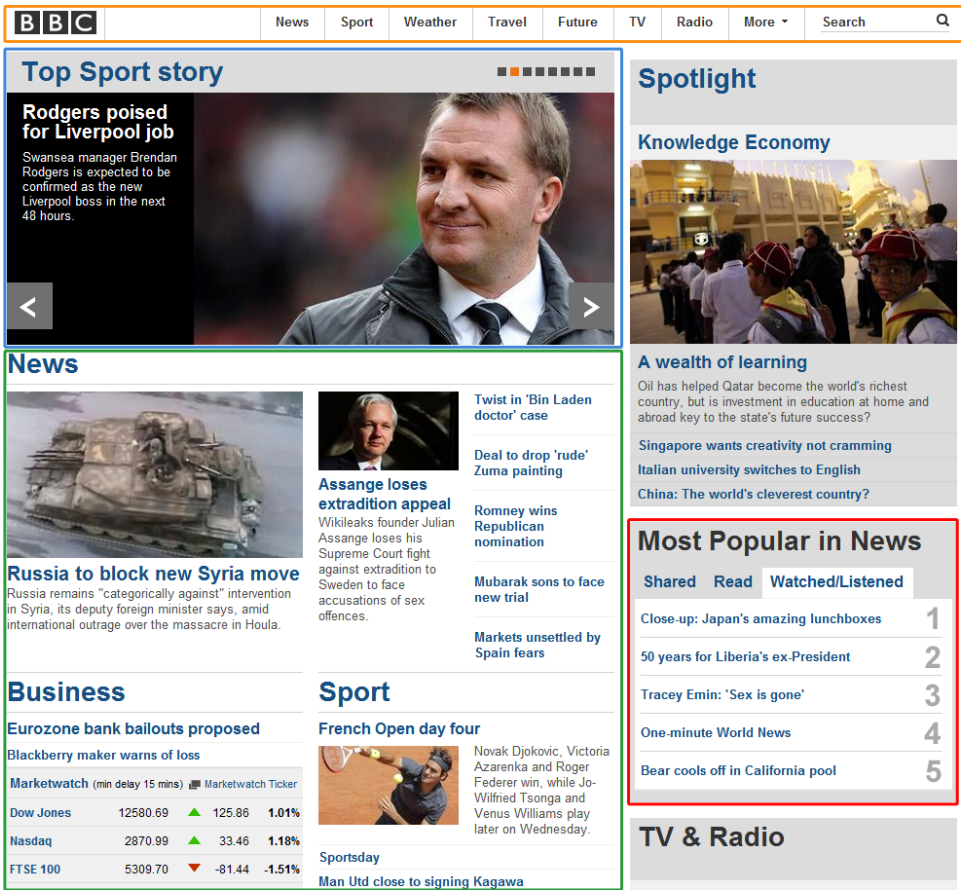


Abbildung 4.3: Startseite von BBC.com vom 19. April 2012

Listing 4.7: HTML-Struktur des Buttons zum Auswählen der vorherigen Schlagzeile

```
1 <div class="nav_left hover" style="width: 123px; cursor: pointer; ">
2   <span class="hide">Left</span><div class="image">
3     <img width="71" height="40" src="http://static.bbci.co.uk/wwhomepage
      -3.5/ic/custom-promo/432-259/promo_alzheimers_3105.jpg">
4   </div>
5 </div>
```

schließen, dass die Funktionalität dieser Elemente nach dem Laden der Seite mit der JavaScript-Bibliothek „jQuery“ festgelegt wird.

Technisch gibt es aktuell keine Möglichkeit, alle derartigen klickbaren Elemente aus einer Internetseite auszulesen. Eine Überprüfung der BBC-Startseite mit dem JavaScript-Werkzeug „Visual Event 2“ [12] lieferte keine registrierten Event-Handler dieser beiden Navigationspfeile.

Sollte BBC zu einem späteren Zeitpunkt WAI-ARIA implementieren, wäre es möglich, die Schaltfläche durch die Rolle „button“ zu identifizieren (vgl. Abschnitt 2.3.4). Die dadurch sehr leicht erkennbaren Schaltflächen könnten dann per JavaScript geklickt werden.

Da die Schlagzeilen auch zeitbasiert umgeschalten werden und die Links zu den Artikeln bereits existieren, wenn die Seite analysiert wird, können diese vom System erfasst werden. Zur Bildung der Sprachbefehle für die Links zu den enthaltenen Artikeln können die Schlüsselwörter verwendet werden, die im folgenden Abschnit aufgezeigt werden.

4.8.3 Links zu Artikeln und Kategorien

Der Bereich mit Links zu Artikeln ist in Abbildung 4.3 grün umrandet und enthält auch Links zu Kategorien. Ein Beispiel der Kategorie ist in Abbildung 4.4 grafisch und in Listing 4.8 als HTML dargestellt. Im aufgelisteten HTML-Code wurden einige nicht relevante Stellen entfernt, um die Übersichtlichkeit zu erhöhen.

Die Kategorie-Container bestehen aus je einem Titel-Link, der auf die Übersichtsseite zum jeweiligen Thema verlinkt und einigen Links zu Artikeln. Der Titel-Link und die Artikel-Links sind einfache Navigationslinks, weshalb die Berechnung ihrer Sprachbefehle ähnlich jener Vorgehensweise ist, die für die Navigationsleiste verwendet wird (vgl. Abschnitt 4.8.1). Sollte es eine Möglichkeit geben, einen Kategorie-Link zu identifizieren, könnten die Schlüsselwörter entsprechend angepasst werden. Die einzige Möglichkeit in diesem Beispiel wäre die Überprüfung, ob ein Link innerhalb einer Überschrift vorkommt, was jedoch nicht zwangsläufig bedeutet, dass der Link ein Titel- oder Kategorie-Link ist.

Zusätzlich könnte der Kontext der Sportnachrichten als Schlüsselwörter in die Sprachbefehle einfließen. Dessen Bestimmung ist jedoch praktisch nicht möglich. Der „div“-Container, der in Abbildung 4.4 dargestellt ist, besitzt zwar die ID „sport_container“, eine Verallgemeinerung für andere Internetseiten kann daraus jedoch nicht abgeleitet werden.

Der Link, der zum Artikel über das Finale des French Open Tennisturniers führt, besteht außerdem aus einem Bild, das den Tennisspieler Novak Djokovic zeigt. Aus dem HTML-Code des Links in den Zeilen 12 bis 17 in Listing 4.8 kann sowohl über das „alt“-Attribut des Links als auch über jenes des verwendeten Bildes ein Kontext zum Inhalt hergestellt werden. Aus diesem Grund ist es möglich, für das Bild einen eigenen Sprachbefehl zu definieren. Ein Beispiel dafür wäre: „show me the picture of novak djokovic please“.

Zu klären wäre in diesem Fall, wie die Erwartungshaltung des Benutzers ist. Da im Sprachbefehl explizit auf das Bild Bezug genommen wird, könnte der Benutzer erwarten, dass das Bild vergrößert dargestellt wird. Die vom System ausgeführte Aktion wäre hingegen das Öffnen des Links.

Abbildung 4.4: Sport-Kategorie der Seite http://www.bbc.com vom 8. Juni 2012

4.8.4 „Most Popular in News“

Der Bereich „Most Popular in News“ ist in Abbildung 4.3 rot umrandet und besteht aus drei Tabs, die über die Links „Shared“, „Read“ und „Watched/Listened“ geöffnet werden. Die Links innerhalb der Tabs führen zu Artikeln und werden somit gleich wie jene in Abschnitt 4.8.3 verarbeitet. Jene Links, die zum Umschalten zwischen den Tabs dienen, sind jedoch keine Navigationslinks und werden daher anders weiterverarbeitet. Die HTML-Struktur des von BBC als „Tabset“ bezeichneten Bereiches ist in Listing 4.9 ersichtlich.

Die drei Links, die durch die CSS-Klasse „tab“ gekennzeichnet sind, besitzen als Ziel ein Rautezeichen, was die Erkennung als reiner Aktions-Link möglich macht (vgl. Abschnitt 4.4.1). Ein „Tabset“ entspricht im Wesentlichen der WAI-ARIA Rolle „tablist“ und deren einzelne Tabs der Rolle „tab“. Mit Hilfe dieser Rollen können die Schlüsselwörter entsprechend angepasst werden.

Die Sprachbefehle zum Umschalten der Tabs entsprechen jenen, die kein „href“-Attribut besitzen oder ein Rautezeichen darin besitzen (vgl. Abschnitt 4.4.1).

4.8.5 Suche

Die Suchfunktion ist in Abbildung 4.3 im orange umrandeten Bereich zu finden. Einen Auszug aus der HTML-Struktur der Seite zeigt Listing 4.10. Wie aus dem HTML-Ausschnitt ersichtlich ist, wurde für das verwendete Formular die WAI-ARIA Rolle „search“ vergeben. Aus diesem Grund ist es möglich, das Textfeld entsprechend zu identifizieren und die Schlüsselwörter für die Suche anzupassen.

Listing 4.8: HTML-Struktur der Sport-Kategorie

```
<div id="sport" class="module">
  <h2 id="sport_title" class="draggable">
    <a rev="..." id="sport_title_link" href="http://www.bbc.co.uk/sport
    /0/" title="Go to ">
      Sport
    </a>
  </h2>
  <!-- Unsichtbarer HTML-Code entfernt -->
  <div id="sport_container" class="container">
    <div class="contentBlocks cbg0">
      <div id="sport_hero_mini" class="hero contentBlock">
        <div class="title title_first">
          <a rev="..." href="..." title="Djokovic &amp; Nadal into dream
     final" class="heroLink">
            <span>
              Djokovic &amp; Nadal into dream final
            </span>
            <img src=".../_60790487_60790486.jpg" alt="Novak Djokovic"
    width="144" height="81">
          </a>
          <p class="summary" id="sport_hero_mini_summary">
            World number one Novak Djokovic beats Roger Federer 6-4 7-5
    6-3 to reach the French Open final for the first time.
          </p>
        </div>
      </div>
      <div id="sport_moreTopStories" class="list contentBlock">
        <ul class="blq-clearfix">
          <li class="first">
            <a rev="sport|homepage|na|r|t|i|text|content" href="...">
              Canadian GP practice
            </a>
          </li>
          <!-- Weitere Links entfernt -->
        </ul>
      </div>
    </div>
  </div>
</div>
```

Listing 4.10 zeigt außerdem, dass für die Suchschaltfläche kein „input“-Element mit dem Typ „submit“ verwendet wird, sondern ein „button“-Element mit dem entsprechenden Typ. Dies hat jedoch keine Auswirkungen auf die Sprachbefehle.

Mögliche Sprachkommandos für die Suche wären:

- „search for [...]“ oder
- „enter the search field“

Der erste Befehl würde direkt eine Suche starten, in dem der in eckigen

Listing 4.9: HTML-Struktur des „Most Popular in News“-Tabset

```
<div id="mostPopular_tabset" class="livestats livestats-tabbed
    contentBlock topTabs">
    <h3 class="first tab selected">
        <a id="mostPopular_tabset_shared_link" href="#">Shared</a>
    </h3>
    <div id="mostPopular_tabset_shared" class="panel selected">
      <!-- Tab-Inhalt entfernt -->
    </div>
    <h3 class="tab">
        <a id="mostPopular_tabset_read_link" href="#">Read</a>
    </h3>
    <div id="mostPopular_tabset_read" class="panel">
      <!-- Tab-Inhalt entfernt -->
    </div>
    <h3 class="tab">
        <a id="mostPopular_tabset_watched_link" href="#">Watched/
    Listened</a>
    </h3>
    <div id="mostPopular_tabset_watched" class="panel">
      <!-- Tab-Inhalt entfernt -->
    </div>
</div>
```

Listing 4.10: HTML-Struktur des Suche auf BBC.com

```
<div id="blq-nav-search">
  <form method="get" action="http://search.bbc.co.uk/search" accept-
    charset="utf-8" id="blq-search-form" role="search">
    <div>
      <!-- Zwei versteckte Felder entfernt -->
      <label for="blq-search-q" class="blq-hide">
        Search term:
      </label>
      <input id="blq-search-q" type="text" name="q" value="" maxlength="
    128" autocomplete="off">
      <button id="blq-search-btn" type="submit">
        <span>
          <img src="...blq-search_grey_alpha.png" width="13" height="13"
     alt="Search">
        </span>
      </button>
    </div>
  </form>
</div>
```

Klammern enthaltene Wert in das Suchfeld eingetragen und das Formular abgeschickt wird. Letzterer Befehl würde den Fokus in das Textfeld setzen, um die Eingabe eines Suchbegriffes zu ermöglichen. Da alle Texteingabefel-

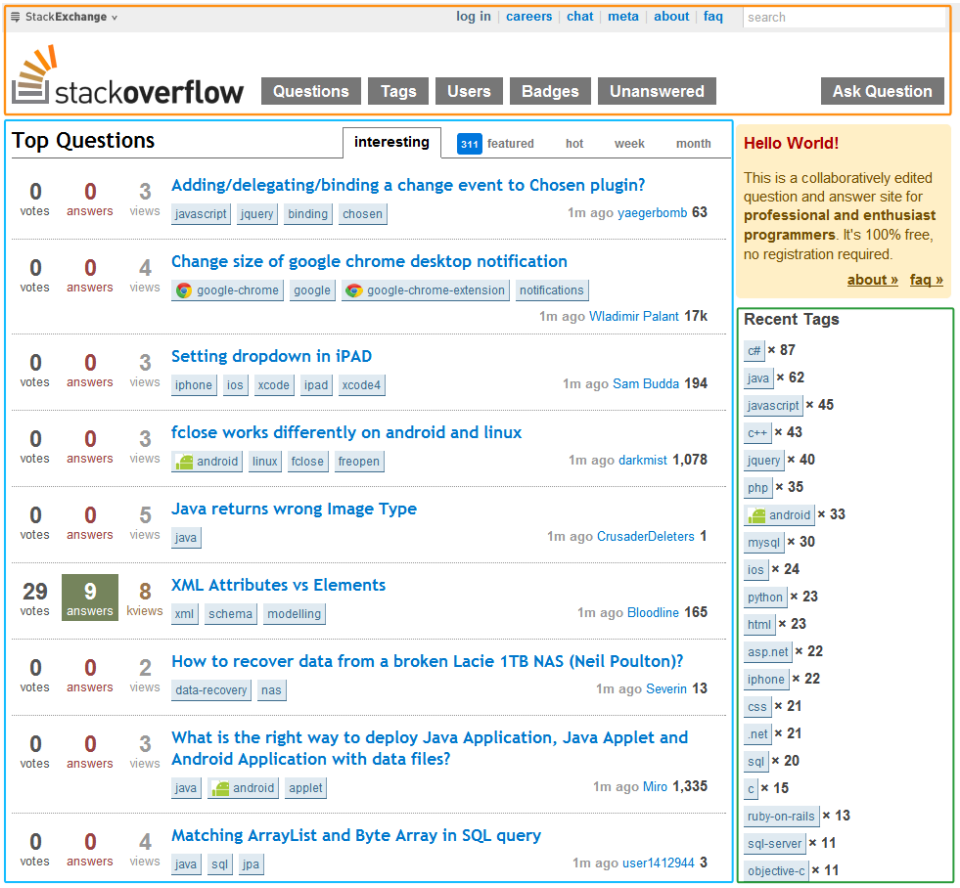


Abbildung 4.5: Startseite von http://www.stackoverflow.com vom 30. März 2012

der mit der Spracheingabefunktion des Google Chrome Browsers ausgestattet werden, kann der Suchtext auch mittels Sprachkommando eingegeben werden.

4.9 Beispiel Stackoverflow

Stackoverflow ist eine „question and answer“-Internetseite, die auf Fragen zum Thema Programmierung spezialisiert ist. Die Fragen werden dabei sowohl von Benutzern gestellt als auch beantwortet. Die Fragen werden „tags“ zugeordnet, die beispielsweise einem bestimmten Thema oder einer Programmiersprache entsprechen. Die Startseite von Stackoverflow ist in Abbildung 4.5 dargestellt und beinhaltet eine farbliche Kennzeichnung der Haupt-Bestandteile der Seite.

Listing 4.11: HTML-Code eines Links zu einer Frage

```
1 <a href="/questions/11028507/programatic-full-import-in-solr-using-java"
       class="question-hyperlink" title="I need to do a full-import with
      Java to a solr server...">
2   Programatic Full Import in Solr using Java
3 </a>
```

Listing 4.12: HTML-Code eines Links zur Kategorie „java“

```
1 <a href="/questions/tagged/java" class="post-tag" title="" rel="tag">
2   java
3 </a>
```

4.9.1 Navigation

Als Navigation können auf der Seite zwei Bereiche angesehen werden, die beide im orange markierten Bereich enthalten sind. Da die Seite im Gegensatz zur Seite der BBC keine WAI-ARIA Rollen beinhaltet, kann die Navigation nicht direkt erkannt werden. Die Sprachbefehle für die beiden Navigationsleisten werden daher mit denselben Schlüsselwörtern generiert, wie jene, die zu den Themen und Fragen auf der Seite führen.

4.9.2 Links zu Fragen und Kategorien

Abermals ist hierbei das Problem, dass Fragen und Kategorien ohne weiteres Wissen über die Internetseite nicht automatisch erkannt werden können. Aus diesem Grund kann auch keine Unterscheidung zu den für die Navigation verwendeten Links gemacht werden. Aus den Listings 4.11 und 4.12 ist ersichtlich, dass eine Unterscheidung nur dann möglich ist, wenn für die verwendeten CSS-Klassen „question-hyperlink“ bzw. „post-tag“ eine Menge von zu verwendenden Schlüsselwörtern festgelegt wird oder weitere semantische Informationen definiert werden.

Abermals würde die Implementierung von „WAI-ARIA“ oder die Verwendung des „article“ Tags aus HTML5 dabei helfen, diese Unterscheidung zu treffen. Dadurch könnten die Fragen identifiziert und mit den für die Kategorie der Internetseite geeigneten Schlüsselwörtern angereichert werden. Dies würde jedoch eine Klassifizierung der Internetseite voraussetzen.

Kapitel 5

Masterprojekt

In diesem Kapitel wird das Masterprojekt beschreiben, das als Grundlage dieser Masterarbeit dient. Dazu werden zuerst die Entwicklungsschritte sowie die verwendeten Technologien beschrieben. Weiters wird auf die zugrunde liegenden Theorien eingegangen, die zur Entwicklung des Projektes notwendig waren.

5.1 Projektbeschreibung

Der Titel des Masterprojektes ist „Sprachsteuerung von Web-Anwendungen“. Es befasst sich mit der Entwicklung eines Prototypen, mit dessen Hilfe eine beliebige Internetseite durch Spracheingaben gesteuert werden kann. Dabei sollen die Sprachkommandos automatisch auf Basis der HTML Struktur der Seite generiert werden. Dabei werden mehrere Faktoren in die Generierung der Sprachbefehle einbezogen, die in Kapitel 4 beschrieben werden. Die Spracheingabe soll ohne die Installation von zusätzlichen Plug-Ins am Computer des Benutzers funktionieren.

5.2 Warum Plug-In frei?

Die Idee von Plug-Ins ist, die Funktionalität eines Browsers bzw. allgemeinen betrachtet einer Software zu erweitern. Dazu wird am Computer eines Benutzers ein Programm installiert, das für bestimmte Inhalte im Browser eingebettet wird und den Inhalt entsprechend darstellt oder verarbeitet. Dieser Ansatz birgt einige Probleme, die im Folgenden kurz erläutert werden [8]:

- Stabilitäts- und Sicherheitsprobleme durch das erhöhte Risiko für Programmfehler in den Plug-Ins.
- Kompatibilitätsprobleme, wenn die Benutzer unterschiedliche Versionen des Plug-Ins installiert haben oder nicht die nötige Berechtigung,

diese zu installieren oder zu aktualisieren.

- Layoutprobleme, wenn sich der für das Plug-In reservierte Bereich mit dynamisch erzeugten HTML-Elementen überschneiden, wie es beispielsweise bei Drop-Down-Menüs passiert. Dies ist dahingehend problematisch, als dass der Bereich, der für das Plug-In reserviert ist, über die anderen HTML-Elemente gelegt wird.

5.3 Technische Voraussetzungen

Eines der Ziele des Projektes war, eine Plug-In freie Lösung für die Sprachsteuerung zu finden. Dieses Ziel wurde zwar grundsätzlich erreicht, für den Betrieb des Prototypen müssen dennoch einige technische Voraussetzungen erfüllt sein. Diese werden nachfolgend erläutert.

5.3.1 Clientseitig

Auf dem Rechner des Clients müssen keine Browser Plug-Ins installiert werden, die Spracheingabe erfordert jedoch den Google Chrome Browser ab Version 11, der spezielle Spracheingabefelder bietet.

5.3.2 Serverseitig

Wie in Anhang A erläutert, wird als zentrale Komponente der Apache HTTPD Server in einer Version ab 2.2 vorausgesetzt. Die weiteren Voraussetzungen sind:

- eine PHP Installation sowie das dazugehörige PHP-Modul für den Apache Server
- eine aktuelle Version des Zend-Frameworks
- ein Ordner für die Index-Dateien des Zend-Lucene Moduls.

Eine detaillierte Installationsanleitung für Ubuntu GNU/Linux ist in Anhang A zu finden.

5.4 Systemarchitektur

Die Architektur des Prototypen, die in Abbildung 5.1 dargestellt ist, besteht aus dem Apache HTTPD Server[1], der inklusive Plug-Ins als zentrale Plattform dient, dem Google Chrome Browser sowie einigen selbst entwickelten PHP-Skripts und JavaScript-Komponenten. Das System kann in zwei Abläufe eingeteilt werden:

- das Aufrufen einer Internetseite (vgl. Abschnitt 5.4.1)
- die Eingabe eines Sprachbefehls (vgl. Abschnitt 5.4.2).

[1] Weitere Informationen unter: http://httpd.apache.org/

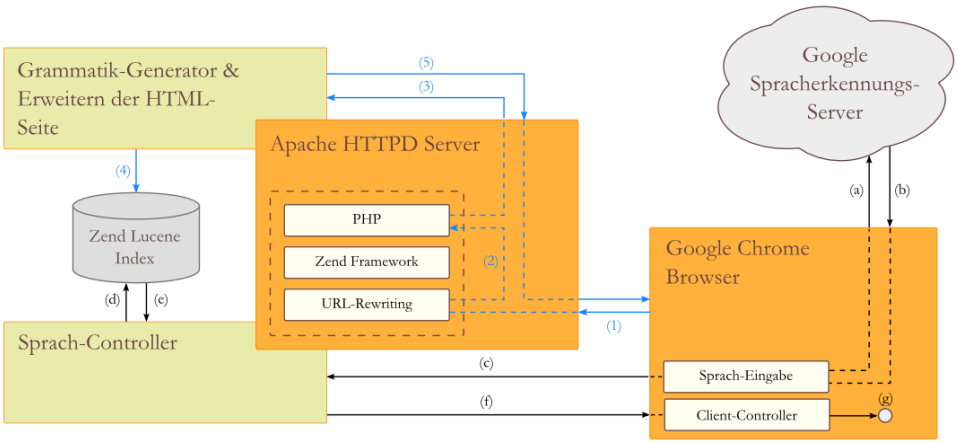


Abbildung 5.1: Grundlegende Systemarchitektur des Masterprojektes

5.4.1 Aufrufen einer Internetseite

Der in Abbildung 5.1 blau dargestellte Ablauf beschreibt das Aufrufen einer Internetseite, die auf dem Apache Server liegt. Ausgehend vom Browser werden die folgenden Schritte durchlaufen, deren Nummerierung jener in der Abbildung entsprechen:

(1) **Aufruf der Seite:** Der Benutzer ruft über seinen Browser eine auf dem Server befindliche Internetseite auf, wodurch HTTP-Anfrage an den Apache Server gesendet wird.

(2) **Interne Umleitung:** Mittels internem „URL-Rewriting“ werden Anfragen für Adressen, die auf „.html“ oder „.htm“ enden, an ein PHP-Skript weitergeleitet.

(3) **Aufrufen des Grammatikgenerators:** Das PHP-Skript ruft den ebenfalls in PHP geschriebenen Grammatikgenerator auf, um die Sprachbefehle für die angefragte Internetseite zu erstellen. Weiters werden bei Bedarf die „id“-Attribute der zur Steuerung relevanten Elemente hinzugefügt, um eine eindeutige Identifizierung dieser zu ermöglichen. Zusätzlich werden jQuery sowie der darauf aufbauende Client-Controller als JavaScript-Dateien sowie das Sprach-Eingabe-Textfeld in die HTML-Seite eingefügt.

(4) **Befüllen des Suchindex:** Die generierten Sprachbefehle werden mittels „Zend Lucene“ (einem Bestandteil des Zend-Frameworks) in den Suchindex gespeichert. Zu den Sprachbefehlen werden weitere Metainformationen wie die bereits erwähnten Werte der „id“-Attribute gespeichert. Diese Metainformationen werden zur Zuordnung von Sprachbefehlen zu HTML-Elementen verwendet und werden unverändert im Suchindex gespeichert.

(5) **Senden und darstellen der Seite:** Die HTML-Seite wird inklusive der Veränderungen an den Browser gesendet und dargestellt.

5.4.2 Eingabe eines Sprachbefehls

Der zweite, in Abbildung 5.1 schwarz dargestellte Ablauf zeigt die Eingabe eines Sprachbefehles und dessen weitere Verarbeitung.

(a) **Aufnahme und Übertragung des Sprachsignals:** Nachdem die Spracheingabe für das Eingabefeld aktiviert und der Sprachbefehl aufgenommen wurde, wird dieser an einen Spracherkennungsserver von Google gesendet.

(b) **Antwort des Spracherkennungsservers:** Die textliche Repräsentation der Aussage wird an den Browser zurückgesendet und in das Spracheingabefeld eingefügt.

(c) **Übertragen des Sprachbefehls an den Apache Server:** Sobald in das Spracheingabefeld ein neuer Wert eingetragen wurde, wird mittels AJAX ein HTTP-Request an den Sprach-Controller gesendet.

(d) **Suchanfrage starten:**Der Sprach-Controller durchsucht den Suchindex nach dem Sprachkommando.

(e) **Verarbeiten der Suchergebnisse:**„Zend Lucene“ liefert eine Liste von Ergebnissen, die nach deren Relevanz sortiert sind. Die Suchergebnisse enthalten jeweils die Metainformationen, die in Abschnitt 5.4.1 erwähnt werden.

(f) **Antwort an den Client:**Die Ergebnisse der Suche sowie die benötigten Steuerinformationen werden im JSON-Format an den Browser zurückgesendet.

(g) **Durchführen der Aktionen:**Der Client-Controller führt anhand der Steuerinformationen die entsprechenden Schritte durch.

Fehlerbehandlung

Im Zuge dieser Schritte können insbesondere dann Fehler auftreten, wenn für den vom Benutzer gesprochenen Text kein Sprachbefehl vorhanden ist oder nur Ergebnisse gefunden werden, deren Relevanz unter einem festgelegten Schwellwert liegt. Für diesen Fall wird eine Fehlermeldung angezeigt, die den Benutzer über die Fehlerursache informiert. Weitere Fehlerquellen sind Störungen der Internetverbindung des Benutzers oder der Spracherkennungsserver, die beide zur Nicht-Verfügbarkeit der Spracheingabe führen.

5.4.3 Grammatikgenerator

Der Grammatikgenerator ist jene Komponente, die sich sowohl um das Auslesen der relevanten Elemente der Internetseite, als auch um die Generierung

Listing 5.1: Beispiele für XPath-Ausdrücke

```
1 //a
2 input[@type="button"]
3 //@onclick
4 //div[@role="navigation"]/a
```

der Sprachgrammatik kümmert, mit der ein bestimmtes Element angesteuert werden kann. Er wird aufgerufen, wenn ein Benutzer eine Internetseite des Servers anfordert. Zusätzlich werden Sprachkommandos für Elemente generiert, die clientseitig mittels JavaScript erstellt werden. Dazu werden die Daten der erstellten Elemente mit einer asynchronen Anfrage an den Server übermittelt. Dieser führt dann dieselben Schritte durch, die auch beim Anfordern einer kompletten Seite durchgeführt werden.

XPath

Das Durchsuchen der HTML-Seite nach relevanten Elementen ist mit Abfragesprache „XPath“ realisiert. Diese kann dazu verwendet werden, ein oder mehrere Elemente aus der Struktur einer HTML-Seite anhand von Attributwerten oder Tag-Namen auszuwählen. In Listing 5.1 sind vier Beispiele für XPath-Ausdrücke aufgelistet, die im Masterprojekt verwendet werden. Zeile 1 zeigt einen Ausdruck, um alle Links in einem HTML-Dokument, unabhängig von deren Attributen oder Kontext zu finden. In Zeile 2 werden alle „input“-Elemente selektiert, deren Attribut „type“ den Wert „button“ besitzt. Mit dem Ausdruck in Zeile 3 werden alle Elemente gefunden, die das „onclick“ Attribut besitzen. Abschließend ist in Zeile 4 ein Ausdruck dargestellt, der alle Links in einem „div“-Element findet, dessen „WAI-ARIA“-Rolle durch das Attribut „role“ mit dem Wert „navigation“ definiert ist (vgl. Abschnitt 2.3.4). Dieser Ausdruck ist zwar prinzipiell korrekt, kann jedoch nicht verwendet werden, da die im Projekt eingesetzte XPath-Implementierung nur HTML4 unterstützt und somit keine benutzerdefinierten Attribute und Elemente erlaubt.

5.4.4 Sprach-Controller

Die Aufgabe des Sprach-Controllers ist es, die vom Benutzer gesendeten Sprachbefehle zu analysieren und das dazugehörige HTML-Element zu finden. Anschließend werden die Steuerungsinformationen an den Client-Controller gesendet, der die entsprechenden Befehle durchführt.

Event	Beschreibung
„click“	Eine Maustaste wurde geklickt
„dblclick“	Eine Maustaste wurde doppelt geklickt
„mousedown“	Eine Maustaste wurde gedrückt
„mouseup“	Eine Maustaste wurde losgelassen
„mouseover“	Der Mauszeiger wurde innerhalb eines Elementes positioniert
„mousemove“	Der Mauszeiger wurde innerhalb eines Elementes bewegt
„mouseout“	Der Mauszeiger wurde aus einem Element heraus positioniert

Tabelle 5.1: Maus-Events der HTML-Spezifikation

5.5 Interaktion mit einer Internetseite

Essentiell für die Steuerung einer Internetseite sind jene Elemente, die eine Interaktion mit der Seite oder einem Server ermöglichen. Historisch betrachtet werden vor allem zwei Eingabegeräte für die Steuerung von Internetseiten verwendet:

1. die Maus, um auf Elemente wie Buttons oder Links zu klicken und
2. die Tastatur, um Textfelder zu befüllen oder z.B. mit der Tabulator-Taste die Elemente der Seite nacheinander anzuwählen bzw. zu fokussieren.

5.5.1 Maus

Die Maus ist auf Desktop-Systemen das am häufigsten verwendete Gerät zur Interaktion mit einer Internetseite. Neben dem Klicken von Maustasten existieren in der HTML-Spezifikation weitere Möglichkeiten, auf Maus-Interaktionen zu reagieren, welche in Tabelle 5.1 aufgelistet sind.

Zusätzlich existieren noch berührungssensitive Eingabegeräte, wie beispielsweise die Bildschirme von aktuellen Smartphones. Diese werden jedoch meist als Maus-Ersatz verwendet, angereichert durch einige Gesten wie das „Wischen“ zum Scrollen einer Internetseite.

Die Interaktion mit der Internetseite erfolgt beinahe ausschließlich durch das Klicken von Elementen, die eine programmierte Funktion, wie z.B. das Navigieren zu einer anderen Internetseite ausführen. Essenziell für die Sprachsteuerung einer Internetseite ist somit das Finden und Verwalten aller klickbaren Elemente sowie das Überwachen der Erstellung von neuen DOM-Elementen mit JavaScript.

Event	Beschreibung
„keypress“	Eine Taste wurde gedrückt und wieder losgelassen
„keydown“	Eine Taste wurde gedrückt
„keyup“	Eine Taste wurde losgelassen

Tabelle 5.2: Tastatur-Events der HTML-Spezifikation

5.5.2 Tastatur

Üblicherweise werden Tastaturen nur dann zur Interaktion mit Internetseiten eingesetzt, wenn Eingabefelder mit Werten befüllt werden. Die verfügbaren Events dazu sind in Tabelle 5.2 aufgelistet. Da üblicherweise alle Funktionen einer Internetseite über die Maus verwendbar sind, wird nicht näher auf Tastatur-Ereignisse eingegangen.

5.6 Überwachen von Veränderungen

Durch die immer häufigere und intensivere Verwendung von JavaScript auf Internetseiten ist es notwendig, Änderungen in der Struktur der HTML-Seite zu überwachen. Dazu zählen jegliche Änderungen an den bereits verarbeiteten Elementen sowie das Erstellen und das Entfernen von Elementen.

Zur Überwachung von Veränderungen des DOM-Baumes stellt JavaScript die „Mutation-Events“ zur Verfügung, welche einige Events für die Überwachung bestimmter Änderungen umfassen. Die jeweiligen Änderungen werden mit einem AJAX-Aufruf an den Sprach-Controller gesendet, der anschließend alle Sprachkommandos, die das entsprechende Element betreffen, aus dem Lucene-Index löscht und neue Kommandos generiert.

5.6.1 Veränderungen bestehender Elemente

Veränderungen von Elementen in HTML-Seiten werden clientseitig mittels JavaScript durchgeführt und können sowohl die Werte von Attributen oder die Werte oder Attribute von Kind-Elementen betreffen. Diese beiden Fälle werden nun näher betrachtet.

Attribut-Werte

Zur Überwachung von Attributwerten kann das JavaScript-Event „DOMAttrModified“ verwendet werden, das alle Änderungen der Attribute eines einzelnen Elementes überwacht. Listing 5.2 zeigt ein Anwendungs-Beispiel für Firefox, Opera und Internet Explorer, wobei bei der Verwendung des Internet Explorers zwischen den Versionen vor 9 und jenen danach unterschieden wird. Das Code-Beispiel wurde basierend auf [33] entwickelt, wurde gekürzt und für die Verwendung von jQuery angepasst.

Listing 5.2: Anwendungsbeispiel für das DOMAttrModified-Event [33]

```
<script type="text/javascript">
$(document).ready(function() {
  if (($.browser.msie && $.browser.version.substr(0,1) < 9) || $.browser
    .opera) {
    $("#elementToWatch").bind("propertychange", onAttrModified);
  } else {
    $("#elementToWatch").bind("DOMAttrModified", onAttrModified, false);
  }
});

function onAttrModified(event) {
  var message = "";
  if (($.browser.msie && $.browser.version.substr(0,1) < 9) || $.browser
    .opera) {
    message += "Something has happened to an attribute of the " + event.
    target.tagName + " element.\n";
  } else {
    message = "The " + event.propertyName + " property of the " + event.
    srcElement.tagName + " element has been changed.";
  }
  alert (message);
}
</script>
```

Das „DOMAttrModified“-Event funktioniert jedoch nicht in WebKit-basierten Browsern, wozu unter anderem Google Chrome und Apple Safari zählen. Da das Masterprojekt auf der Verwendung von Google Chrome aufbaut, musste eine andere Möglichkeit gefunden werden, die Attribut-Werte zu überwachen. Mit Hilfe der „setInterval“-Funktion wird eine JavaScript-Funktion in einem bestimmten Intervall wiederholt ausgeführt, bis die Ausführung explizit beendet wird. Die Funktion, die im Intervall von 50 Millisekunden ausgeführt wird, ist in Listing 5.3 dargestellt. Der Original-Code aus [19] überprüft übergebene CSS-Werte nach Veränderungen und wurde daher angepasst, um Veränderungen von Attribut-Werten zu überwachen. Dies geschah durch das Austauschen aller „.css()“-Funktionsaufrufe durch „.attr()“.

Das Codebeispiel in Listing 5.3 definiert ein jQuery-Plugin, das auf Elemente angewendet werden kann. Gegenüber dem Originalcode wurde das Beispiel etwas gekürzt und beinhaltet nicht die im Google Chrome Browser unverfügbaren Events „propertychange“ und „DOMAttrModified“.

5.6.2 Erstellung und Entfernung von Elementen

Zum Überwachen der Erstellung und Entfernung einzelner Elemente aus dem DOM-Baum der Internetseite existieren einige JavaScript-Events:

Listing 5.3: Codebeispiel einer „watch“-Funktion für jQuery

```
1 $.fn.watch = function(props, callback, timeout){
2     if(!timeout)
3         timeout = 10;
4     return this.each(function(){
5         var el      = $(this),
6             func    = function(){ check.call(this, el) },
7             data    = { props:  props.split(","),
8                         func:   callback,
9                         vals:   [] };
10        $.each(data.props, function(i) {
11              data.vals[i] = el.attr(data.props[i]);
12        });
13        el.data(data);
14        setInterval(func, timeout);
15    });
16    function check(el) {
17        var data    = el.data(),
18            changed = false,
19            temp    = "";
20        for(var i=0;i < data.props.length; i++) {
21            temp = el.attr(data.props[i]);
22            if(data.vals[i] != temp){
23                data.vals[i] = temp;
24                changed = true;
25                break;
26            }
27        }
28        if(changed && data.func) {
29            data.func.call(el, data);
30        }
31    }
32 }
```

- **„DOMNodeInserted“** wird ausgelöst, nachdem ein Element in den DOM-Baum eines anderen Elementes eingefügt wurde;
- **„DOMNodeRemoved“** wird ausgelöst, nachdem ein Element aus dem DOM-Baum des überwachten Elementes entfernt wurde;
- **„DOMNodeInsertedIntoDocument“** wird einmalig ausgelöst, wenn ein bestimmtes Element in den DOM-Baum eingefügt wurde;
- **„DOMNodeRemovedFromDocument“** wird einmalig ausgelöst, nachdem ein bestimmtes Element aus dem DOM-Baum entfernt wurde.

Der Unterschied zwischen den ersten und letzten beiden Events dieser Auflistung besteht in der Art der Anwendung. Die ersten beiden Events werden dazu verwendet, Änderungen im DOM-Baum eines Elements zu überwachen. Wenn für diese Events ein Listener auf das „body“-Element der Internetseite gesetzt wird, können alle Veränderungen in dessen DOM-Baum

überwacht werden.

Listener für die letzten beiden Events werden hingegen auf ein bestimmtes, in JavaScript erstelltes Element gesetzt und werden einmalig ausgeführt, wenn dieses bestimmte Element in den DOM-Baum der Internetseite eingefügt wird.

Die Anforderung für das Masterprojekt ist es, das Erstellen und Entfernen von Elementen in grundsätzlich unbekannten Internetseiten zu überwachen. Da das System nichts über die zu erstellten oder entfernten Elemente weiß, können die auf ein bestimmtes Element angewandten Events „DOMNodeInsertedIntoDocument“ und „DOMNodeRemovedFromDocument“ nicht verwendet werden.

Problematisch ist der Einsatz der beiden Events aufgrund deren möglicher mehrfachen Auslösung. Sie können auch dann ausgelöst werden, wenn ein Element an eine andere Stelle verschoben wurde. Derartige Situationen treten dann auf, wenn – wie benötigt wird – der gesamte DOM-Baum der Internetseite auf Änderungen überwacht wird. Es müsste daher eine Möglichkeit geschaffen werden, eine Mischung der ersten beiden und letzten beiden Events der obigen Auflistung zu verwenden. Eine weitere Möglichkeit zur Einschränkung der Überwachung sind die sogenannten „Live-Regions“ aus WAI-ARIA. Dadurch müssten nur jene Bereiche der Internetseite überwacht werden, die verändert werden können.

Afugrund dieser Problematiken ist das Überwachen von Veränderungen im Masterprojekt nicht aktiviert. Der dazugehörige Quellcode ist in den Dateien „domchange.php“ und „controller.js“ enthalten.

5.7 Volltextsuche

Im Masterprojekt wird die PHP-Portierung „Zend Search Lucene“ der Java-Bibliothek „Apache Lucene“ verwendet. Dieser Abschnitt beschreibt die Entscheidungsfindung, die zur Auswahl der PHP-basierten Lösung geführt haben. Nach einer Vorauswahl wurden „MySQL“, „Apache Lucene“, „Apache Solr“ sowie „Zend Search Lucene“ näher betrachtet und basierend auf den folgenden Kriterien bewertet:

- Die Software sollte unter einer freien Lizenz stehen, um sowohl eine kostenlose Verwendung zu gewährleisten, als auch einen Einblick in die Funktionsweise zu ermöglichen.
- Die Engine sollte eine einfach zu verwendende Programmierschnittstelle bieten, um die Einarbeitungs- und Integrationszeit zu minimieren.
- Die Abhängigkeiten zu anderen Software-Komponenten sowie die Systemanforderungen sollten möglichst gering sein.
- Für die einfache Anbindung an das bestehende System sollte die Engine nach Möglichkeit in PHP implementiert sein.

- Die Suchergebnisse sollten nach Relevanz bzw. Ähnlichkeit zum Suchtext sortiert werden können.
- Der Konfigurationsaufwand sollte möglichst gering sein.

5.7.1 MySQL

Eine Möglichkeit ist es, die Sprachbefehle in einer MySQL-Datenbank abzulegen und Suchanfragen per SQL zu senden. Zu diesem Zweck bietet MySQL die Befehle „MATCH AGAINST“ sowie „LIKE“. Listing 5.4 zeigt eine Suchanfrage mittels „MATCH AGAINST“. Dabei wird die Spalte „command_text“ mit den Werten in der nachfolgenden Klammer verglichen. Das Ergebnis dieser Abfrage ist eine absteigend nach Relevanz („score“) sortierte Liste von Datensätzen, die mindestens eines der Suchwörter („go to sports news“) enthält. Das abgefrage Feld „html_id“ enthält die eindeutige HTML-ID des Elementes, aus dem der Sprachbefehl generiert wurde und dient zum Ansprechen des Elements durch den Client-Controller. Durch das zweimalige Verwenden der „MATCH AGAINST“-Klausel mit den selben Werten entsteht kein Performance-Nachteil, *„da der MySQL-Optimierer bemerkt, dass die beiden MATCH()-Aufrufe identisch sind und den Code für die Volltextsuche insofern nur einmal aufruft“* [29].

Listing 5.4: Beispiel einer SQL-Suchanfrage mit „MATCH AGAINST“

```
1 SELECT html_id, MATCH(command_text) AGAINST ('go to sports news') AS
      score
2 FROM commands
3 WHERE MATCH (command_text) AGAINST ('go to sports news')
4 ORDER BY score DESC;
```

In Listing 5.5 ist eine Abfrage mit dem „LIKE“-Befehl dargestellt. Um die Toleranz der Suche zu erhöhen, können Wildcard-Zeichen verwendet werden, die für eine bestimmte Anzahl anderer Zeichen stehen. Ohne Wildcards müsste der Satz „go to sports news“ exakt mit einem gespeicherten Sprachbefehl übereinstimmen. Das Prozentzeichen steht für eine beliebige Anzahl von Zeichen (auch 0), die an der Stelle der Wildcard vorkommen können. Die Suchabgrafe in Listing 5.5 erlaubt somit eine beliebige Anzahl an Zeichen vor und nach dem Suchstring sowie zwischen den einzelnen Suchwörtern. Nicht möglich mit dieser Methode ist die Berechnung der Relevanz der einzelnen Suchtreffer.

Listing 5.5: Beispiel einer SQL-Suchanfrage mit „LIKE“

```
1 SELECT id,html_id
2 FROM commands
3 WHERE command_text LIKE '\%go\%to\%sports\%news\%';
```

Bewertung anhand der Kriterien

MySQL ist ein Open-Source Projekt, das unter der „GNU General Public License“ (GPL) und einer kommerziellen Lizenz veröffentlicht wird. Die Verwendung im nicht-kommerziellen Bereich ist somit kostenfrei. Der Installations- und Wartungsaufwand ist für eine Basisinstallation sehr gering. Für die Verwendung von MySQL aus PHP stehen spezielle Funktionen zum Verbinden zur Datenbank sowie zum Senden von Abfragen bereit. Durch die weite Verbreitung von PHP und MySQL ist die Dokumentation sowie die Anzahl an Tutorials im Internet sehr hoch. Der Nachteil für MySQL ist die Verwendung von sehr starren Datenmodellen, die für eine Erweiterung der zu speichernden Daten nicht sehr komfortabel ist. Zudem sind die Möglichkeiten zur Anpassung der Suchparameter sehr gering.

5.7.2 Apache Lucene

Die in Java geschriebene Bibliothek Apache Lucene steht unter der „Apache License“, welche eine kommerzielle Nutzung der Software erlaubt. Durch die Inkompatibilität der Programmiersprachen PHP und Java ist eine direkte Verwendung der Bibliothek in PHP nicht möglich. Eine Möglichkeit ist die Verwendung der PHP-Funktion „shell_exec“, die es erlaubt, einen beliebigen Systembefehl auszuführen. Dieser Systembefehl kann dazu verwendet werden, ein Java-Programm aufzurufen, welches als Ausgabe die Suchergebnisse als Array oder im JSON-Format zurückliefert. Diese Datenstruktur, welche als Text in einer PHP-Variable gespeichert wird, kann interpretiert und weiterverwendet werden. Ein Beispielaufruf eines Java-Programms in PHP ist in Listing 5.6 dargestellt. Die Variable „$output“ enthält nach dem Funktionsaufruf die komplette Ausgabe des Java-Programms. Das aufgerufene Java-Programm muss über eine „main“-Methode verfügen und über Kommandozeilenargumente konfigurierbar sein. Die Implementierung sowie die Definition der Argumente des Java-Programms sind hierbei beliebig gewählt und müssen entsprechend an das verwendete Java-Programm angepasst werden. Weiters enthält Listing 5.6 ein Beispiel für einen Aufruf des Java-Programms zur Indizierung eines neuen Sprachkommandos.

Listing 5.6: Aufruf einer Java-Funktion aus PHP

```
1 // Suchanfrage
2 $query = "go to sports news";
3 $output = shell_exec("java -jar my-lucene-tool.jar -search $query");
4
5 // Indizieren von neuen Inhalten
6 $link_html_id = "abcde";
7 $command = "go to sports news";
8
9 $return_value = shell_exec("java -jar my-lucene-tool.jar -index -id=
      $link_html_id -command=$command1");
```

Die beschriebene Methode bietet zwar Zugriff auf die ursprüngliche Java-Version von Apache Lucene, die im Vergleich zur PHP-Portierung performanter ist, insgesamt überwiegen jedoch die Nachteile, die nachfolgend aufgelistet sind:

- Direkte Kommunikation mit dem System, wodurch die PHP-Anwendung nicht mehr ohne weiteres auf jeder Plattform verwendet werden kann
- Bei Veränderungen in der Schnittstelle zum aufgerufenen Programm muss der Systemaufruf manuell angepasst werden
- Kompatibilitätsprobleme zwischen der am System installierten Java Laufzeitumgebung und dem Java-Programm bzw. der Lucene-Bibliothek
- Erschwerte Fehlerbehandlung
- Erhöhter Wartungsaufwand durch ein zusätzliches Java-Programm

5.7.3 Apache Solr

Apache Solr ist ein auf der Apache Lucene Bibliothek aufbauender Such-Server, der als Anwendung innerhalb eines Servlet-Containers wie Tomcat oder Glassfish betrieben wird. Der Server bietet *„REST-like HTTP/XML and JSON APIs that make it easy to use from virtually any programming language“* [34]. Da der Zugriff über HTTP erfolgt, kann Solr aus jeder netzwerkfähigen Programmiersprache verwendet werden. Neben den Funktionen der zugrundeliegenden Apache Lucene Bibliothek, kann Solr auch in Clustern verwendet werden, ist somit sehr gut skalierbar und bietet „Hit Highlighting“[2] sowie Unterstützung für Dokumentenformate wie Microsoft Word und PDF.

Aufgrund der Tatsache, dass die Verwendung von Apache Solr keine in Java geschriebenen Komponenten benötigt, ist Solr für den Einsatz in PHP-Anwendungen besser geeignet als Lucene. Ein Nachteil ist die Notwendigkeit eines zusätzlichen Java Applikations-Servers wie z.B. Tomcat, um den Such-Server ausführen zu können. Solr eignet sich insbesondere für Anwendungen mit vielen gleichzeitigen Benutzern und sehr großen Suchindizes.

5.7.4 Zend Search Lucene

Zend Search Lucene ist wie in Abschnitt 2.4.2 beschrieben, eine an Apache Lucene angelehnte und großteils kompatible Volltextsuche, die vollständig in PHP umgesetzt ist. Diese Version der Lucene-Suche ist zwar langsamer als die in Java geschriebene, ist jedoch für das vorliegende Projekt am besten

[2] Unter „Hit Highlighting“ versteht man das hervorheben der Suchbegriffe in Suchergebnissen

geeignet. Dies ist vor allem in der direkten Verwendbarkeit der Funktionen in den in PHP geschriebenen Teilen des Systems begründet.

Weiters ist der Funktionsumfang der PHP-Version ausreichend, da im aktuellen Entwicklungsstadium des Masterprojektes keine fortgeschrittenen Funktionen verwendet werden.

Kapitel 6

Praktische Versuche

In diesem Kapitel wird ein Versuchsaufbau sowie die Ergebnisse einiger praktischer Versuche mit mehreren Testpersonen beschrieben, um die Funktionsweise des Systems und die entwickelten Theorien zur Generierung der Sprachbefehle zu testen.

6.1 Versuchsaufbau

Zur Durchführung des Versuches wurde ein handelsüblicher Laptop mit einem Ubuntu GNU/Linux 12.04 LTS 64 Bit Betriebssystem verwendet, als Mikrofon ein „Philips Speech Mike Pro LFH-7274“. Als Web-Browser diente Google Chrome in der Entwicklerversion „20.0.1132.3 dev“.

6.2 Verwendete Internetseiten

Für die Tests wurden zwei verschiedene Internetseiten herangezogen, die am 15. Mai 2012 mit dem Programm „wget“[1] bis zu einer Linktiefe von 2 heruntergeladen wurden. Der dafür verwendete Befehl lautet:

```
1 wget -r -l2 http://www.bbc.com
```

Die verwendeten Optionen werden im „wget“-Benutzerhandbuch[2] beschrieben. Weiters wurden bei absoluten Links die Domain entfernt und mit „localhost“ ersetzt, um auf die lokal gespeicherte Seite zu verweisen. Dadurch wurde es möglich die Vorwärts- und Rückwärts-Navigation zu testen. Weitere Änderungen wurden an den einzelnen Internetseiten nicht durchgeführt.

6.2.1 BBC.com

Die Internetseite von BBC ist eine typische Nachrichtenseite, die auf der Startseite eine Übersicht über aktuelle Themen bietet. Außerdem ist über ei-

[1] GNU Wget 1.13.4 built on linux-gnu

[2] http://www.gnu.org/software/wget/manual/wget.html

ne Navigationsleiste die Möglichkeit gegeben, zu speziellen Themen zu wechseln. Für die Tests ist diese Internetseite bestens geeignet, da sie sehr gut strukturiert ist und Links sowohl aus Text, als auch aus Bildern bestehen.

Untersucht wurde außerdem, ob sich die verwendeten Schlüsselwörter zwischen den Links der Navigationsleiste und den Links zu einzelnen Artikeln unterscheiden.

6.2.2 Stackoverflow.com

Die Seite Stackoverflow.com bietet als Frage-Antwort-Plattform jedem Benutzer die Möglichkeit, Fragen zur Computerprogrammierung zu stellen. Aus diesem Grund beinhalten die darin enthaltenen Links sehr viele Abkürzungen und Sonderzeichen, wodurch deren Ersetzung in den Sprachbefehlen getestet werden kann.

Weiters konnte auf dieser Seite getestet werden, ob die verwendeten Schlüsselwörter bei der Auswahl von einzelnen Fragen sowie den Kategorien und der Navigationsleiste unterschiedlich sind .

6.3 Versuchsablauf

Der erste Schritt war die Einschulung der Versuchspersonen in die Funktionsweise des Systems. Dabei wurde erklärt, wie ein Sprachbefehl an das System gesendet werden kann und welche Funktionen das System untertützt. Die Versuche wurden mit fünf Personen durchgeführt, die alle als technikaffin zu bezeichnen sind.

Der zweite Schritt bestand darin, die Teilnehmer dazu aufzufordern, verschiedene Sprachbefehle ohne Vorgaben zu testen, um sich mit dem System vertraut zu machen. Neben den gesprochenen Sätzen wurde dabei notiert, ob die vom Tester gewünschte Aktion vom System durchgeführt wurde. Sollte die durchgeführte Aktion nicht die gewünschte gewesen sein, wurde dies samt der erwarteten Aktion notiert. Dabei wurde auch darauf geachtet, ob das Spracherkennungssystem die Wörter korrekt erkannte. Der Versuch wurde auf beiden in Abschnitt 6.2 erwähnten Internetseiten durchgeführt.

Für den Fall, dass Sprachbefehle verwendet wurden, die einen offensichtlichen Zweck hatten und nicht vom System unterstützt werden, wurden die Tester darauf aufmerksam gemacht. Anschließend wurden die Tester zur erwarteten Funktionsweise des Befehles befragt.

Die Erkenntnisse der Tests sind in zwei Bereiche einzuteilen:

1. seitenübergreifende Erkenntnisse, die allgemein gültig sind und
2. Erkenntnisse, die für je eine der beiden verwendeten Internetseiten gültig sind.

6.4 Seitenübergreifende Erkenntnisse

In diesem Abschnitt werden Erkenntnisse beschrieben, die unabhängig zur verwendeten Internetseite stehen. Dazu zählen neben der Struktur der verwendeten Sprachbefehle auch Funktionen, die von den Testpersonen angesprochen wurden, vom System jedoch nicht unterstützt werden.

6.4.1 Verwendete Sprachbefehle

Das Hauptaugenmerk der praktischen Versuche lag auf der Untersuchung der verwendeten Sprachbefehle. Dabei wurde untersucht, welche Schlüsselwörter und welcher Teil eines Link-Textes verwendet und wie die Befehle für Bild-Links gebildet wurden.

Schlüsselwörter

Die Schlüsselwörter, die zur Navigation auf der Internetseite verwendet wurden, waren beinahe ausschließlich jene, die in Abschnitt 4.4.1 aufgelistet sind. Sehr selten wurden Varianten wie „read [...] article“ verwendet. Insgesamt war die Anzahl der verwendeten Schlüsselwörter sehr gering, was es sehr schwer macht, Rückschlüsse auf den Kontext eines Links zu ziehen, um die in Abschnitt 4.4.1 aufgestellten Thesen zu bestätigen oder zu widerlegen.

Verwendete Teile des Link-Textes

Bei reinen Text-Links konnte festgestellt werden, dass insbesondere die beschreibenden Teile eines Link-Textes als Sprachbefehl verwendet wurden. Die am häufigsten verwendeten Teile des Link-Textes waren Nomen, was anhand eines Beispieles verdeutlicht werden kann.

Die in Abbildung 4.3 dargestellte Startseite von http://www.bbc.com zeigt die „Top Sport Story“ mit der Schlagzeile „Rodgers poised for Liverpool job“. Die Testpersonen kürzten hierbei den Link-Text derart ab, dass er nur mehr aus Hauptwörtern bestand. Die verwendeten Sprachbefehle entsprachen in etwa dem folgenden: „go to rodgers liverpool job“. Wie daraus ersichtlich ist, wurde sowohl das Verb „poised“ (engl. für „bereit“) als auch die Präposition „for“ ignoriert. Diese Vorgehensweise konnte bei allen fünf Testpersonen beobachtet werden.

Basierend auf diesem Ergebnis könnten Optimierungen am System vorgenommen werden. Lucene bietet eine Abfragesprache, die eine Filterung oder die Gewichtung von Teilen des Suchbegriffes ermöglicht. Würde man die Gewichtung aller Nomen und Verben bei der Abfrage erhöhen, könnte möglicherweise die berechnete Relevanz von Suchergebnissen erhöht werden. Dies müsste jedoch durch weitere Versuchsreihen getestet werden und ist nicht Teil dieser Arbeit.

Bildbeschreibungen

Zur Navigation wurden von den Testpersonen sehr selten Links verwendet, deren klickbarer Bereich aus Bildern besteht. Auf der Seite der BBC enthielten „Top News Stories“ ein Bild sowie einen Link zu einem Artikel über das Apple iPad[3]. Der dabei verwendete Sprachbefehl lautete „show me picture ipad“. Durch die Verwendung dieses Sprachbefehls ist nicht klar, ob die Testperson nur das Bild vergößert sehen oder den dazugehörigen Artikel lesen wollte. Dies ist jedoch hinfällig, da der verlinkte Artikel meist ein größeres Bild zu dem Thema enthält.

6.4.2 Scrollen

Das Scrollen auf einer Seite wurde von allen fünf Testpersonen versucht. Das System konnte zum Testzeitpunkt die entsprechende Funktion nicht ausführen. Zum Scrollen auf einer Seite wurden zwei Arten von Befehlen verwendet, welche in Tabelle 6.1 aufgelistet sind.

Die Befehle ohne eine Scroll-Weite, könnten vom System direkt verwendet werden, für jene die keine Scroll-Weite beinhalten, muss ein sinnvoller Wert gefunden werden.

Ohne definierte Scroll-Weite	Fest definierte Scroll-Weite
„scroll down“	„scroll to the bottom of the page“
„scroll up“	„go to the top of the page“

Tabelle 6.1: Zum Scrollen auf einer Seite verwendete Sprachbefehle

Die erwartete Scroll-Weite war bei allen Testpersonen ähnlich:

- Zwei Testpersonen erwarteten eine Scroll-Weite, die exakt der Höhe der sichtbaren Fläche des Browsers entspricht, während
- die drei restlichen Testpersonen eine etwas geringere Höhe erwarteten, was der Orientierung auf der Seite dient.

Eine der Testperson merkte außerdem an, dass die Scroll-Bewegung nicht ruckartig, sondern fließend sein soll. Dies sollte ebenfalls der Orientierung auf der Internetseite zu Gute kommen.

Insgesamt ist jene Variante zu bevorzugen, bei der ein Teil des zuvor angezeigten Inhaltes nach dem Scrollen am oberen Rand der Seite sichtbar ist. Zu begründen ist dies mit der Tatsache, dass einzelne Texte oder Bilder unter Umständen abgeschnitten dargestellt werden. Abbildung 6.1 zeigt ein entsprechendes Beispiel. Der rot hervorgehobene Bereich stellt dabei den sichtbaren Bereich vor (linke Seite) und nach dem Scrollen dar. Anhand der Abbildung ist ersichtlich, dass nach dem Scrollen nur der untere Teil des Bildes sichtbar ist.

[3] iPad ist eine Marke von Apple Inc.

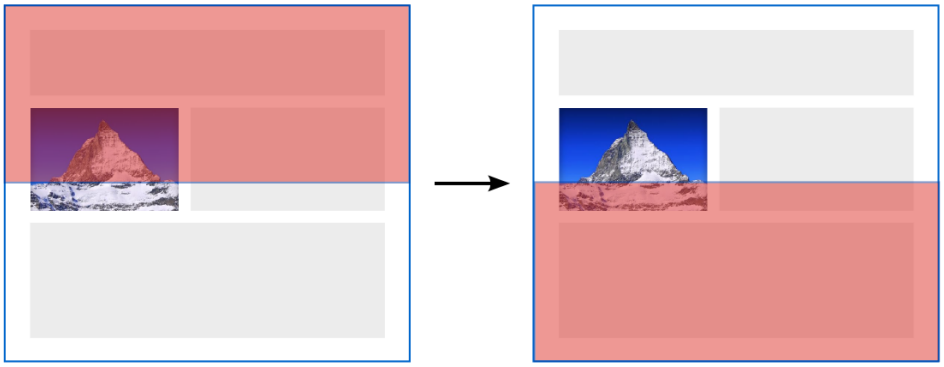


Abbildung 6.1: Problematik bei Verwendung der gesamten Seitenhöhe als Scroll-Weite (Bild: „The Matterhorn“ von http://www.freedigitalphotos.net/

Aus dieser Problematik lässt sich schließen, dass die Scroll-Weite derart gewählt werden sollte, dass in etwa zwei bis drei Zeilen des zuvor sichtbaren Bereiches nach dem Scrollen noch angezeigt werden.

6.4.3 Suche

Beide für die Versuche verwendeten Internetseiten bieten eine Suchfunktion. Die Eingabe erfolgt bei beiden Seiten über ein Texteingabefeld am rechten Rand des Kopfbereichs, wie die Abbildungen 4.3 und 4.5 zeigen.

In Abschnitt 4.8.5 werden zwar Sprachbefehle für die Verwendung der Suchfunktion aufgelistet, diese konnten jedoch aufgrund von technischen Einschränkungen nicht getestet werden. Die Internetseite der BBC verwendet zwar korrekt die WAI-ARIA-Rolle „search“ für das Suchformular, derartige Rollen können jedoch nicht von dem im Masterprojekt eingesetzten HTML-Parser verarbeitet werden. Dies resultiert aus der Tatsache, dass dieser nur HTML4 versteht und die WAI-ARIA Rollen nicht Teil dieses Standards sind. Unabhängig von der technischen Realisierbarkeit wurde mit den Testpersonen besprochen, welche Sprachbefehle sie verwenden würden und wie das erwartete Verhalten der Suchfunktion sei.

Die verwendeten Sprachbefehle waren Variationen der folgenden beiden:

1. „search for [...]“
2. „search site for [...]“

Dies beiden Befehle unterscheiden sich auch im erwarteten Verhalten. Während der erste Befehl die auf der Seite angebotene Suche verwenden sollte, war die erwartete Funktionsweise des zweiten die Suche innerhalb einer Seite. Das in Abschnitt 4.8.5 beschriebene Fokussieren des Suchfeldes auf der Seite mit anschließender manueller Eingabe des Suchbefehles wurde

von keiner Testperson verwendet.

Die Suche innerhalb einer Seite wurde von zwei Testpersonen bevorzugt. Dabei sollte die Funktionsweise ähnlich der vom Browser angebotenen Suchfunktion sein, welche mit der Tastenkombination „Strg + F“ aufgerufen werden kann. Um diese Funktion entsprechend Nachbilden zu können, wären die folgenden Erweiterungen am System nötig:

- Speicherung des aktuellen Suchkontextes, um eine Suche ausgehend von einem bestimmten Punkt der Seite zu ermöglichen
- Zusätzliche Sprachbefehle, die eine vorwärts- bzw. rückwärtsgerichtete Suche ausgehend vom Suchkontext starten
- Korrekte Aufteilung des Sprachbefehles in den eigentlichen Befehl und den Suchtext
- Eine Möglichkeit, die gefundenen Treffer innerhalb einer Seite farblich hervorzuheben
- Eine clientseitig funktionierende Suchfunktion

Die Verwendung der von der Internetseite angebotenen Suchfunktion wurde von drei Testpersonen gegenüber der seiteninternen Suche bevorzugt.

6.4.4 Fenster und Tabs

Neben den bereits vorhandenen Funktionen zur Steuerung des Browsers, wie dem Ausdrucken der Internetseite, wurden von einer Testperson die Befehle „open [...] in a new tab“ sowie „open [...] in a new window“ verwendet. Die Platzhalter entsprechen hierbei dem Text des zu öffnenden Links.

Das Öffnen eines Links in einem neuen Tab oder Fenster kann grundsätzlich mit JavaScript durchgeführt werden. Die dazu verwendete Funktion ist „window.open“[4], ob sie einen neuen Tab oder ein neues Fenster öffnet, ist jedoch nicht definiert und hängt von den persönlichen Einstellungen des Benutzers ab.

6.4.5 Markieren von Text

Das Markieren von Text wurde von einer Testperson angesprochen. Mit dem Sprachbefehl „select [...]“ sollte der Text innerhalb des Platzhalters markiert werden. Dieser markierte Text sollte dann entweder kopiert oder für eine Google-Suche verwendet werden.

Basierend auf der Suche innerhalb einer Seite könnten Sprachbefehle zum Kopieren und Starten einer Google-Suche definiert werden, die den von der Suche markieren Text verwenden. Dies wäre jedoch sehr umständlich zu verwenden, insbesondere wenn der zu markierende Text einmalig auf der Internetseite ist. Bei der Verwendung einer Speziellen Software auf dem Rechner des Benutzers könnte eine Google-Suche direkt angestoßen werden.

[4] http://de.selfhtml.org/javascript/objekte/window.htm#open

6.4.6 Aktivieren eines Seiten-Bereiches

Um die Auswahl der klickbaren Elemente einzugrenzen, schlug eine Testperson das Aktivieren einzelner Bereiche einer Internetseite vor, um die darin enthaltenen Elemente zu aktivieren.

Dies wäre beispielsweise der türkis markierte Bereich in Abbildung 4.5. Basierend auf dieser Markierung könnte die Suche auf die in diesem Bereich enthaltenen Elemente eingeschränkt werden. Zusätzlich wäre denkbar, die nicht verwendeten Bereiche der Internetseite ähnlich einer Lightbox abzudunkeln. Dies würde vorrangig der Benutzbarkeit der Funktion dienen.

Diese Funktion würde voraussetzen, dass die Internetseite in mehrere Bereiche sinnvoll eingeteilt werden kann. Eine Einteilung in Kopf- und Seitenbereiche sowie den Hauptbereich wäre möglich, die Schlüsselwörter zum Ansteuern dieser Bereiche wären jedoch nicht immer klar. Auf der Seite Stackoverflow könnte beispielsweise der in Abbildung 4.5 grün markierte Bereich der Tags hervorgehoben werden. Zur Definition von Sprachbefehlen für diesen Bereich müsste ein Name existieren, der ihn eindeutig identifiziert. In diesem Fall wäre dies die Überschrift „Recent Tags“, die jedoch rein aus der HTML-Struktur nicht direkt ersichtlich ist.

6.4.7 Google-Spracherkennung

Grundsätzlich lieferte die Google-Spracherkennung sehr genaue Ergebnisse zu den Sprachbefehlen. Probleme wurden insbesondere dann festgestellt, wenn die vorhandene Internetverbindung sehr langsam ist.

Außerdem war die Erkennung von Abkürzungen häufig nicht sehr genau, wie im vorherigen Abschnitt bereits beschrieben wurde.

6.5 Erkenntnisse auf BBC.com

Die erste Erkenntnis auf dieser Internetseite war, dass die Sprachsteuerung besser funktioniert als auf der Seite „Stackoverflow.com“. Die am häufigsten verwendeten Links waren jene, die aus reinem Text bestanden sind, Bild-Links wurden, wie schon beschrieben, sehr selten verwendet.

Kleinere Probleme konnten wiederum mit der Spracherkennung festgestellt werden, welche die Namen von Personen und Orten bzw. Ländern manchmal nicht richtig erkannte. Dies ist jedoch aufgrund der sehr verschiedenen Aussprache von Namen in unterschiedlichen Sprachen und Dialekten nicht verwunderlich. Vor allem auch deshalb, weil die Google-Spracherkennung keinen speziellen Einsatzzweck besitzt.

Die Navigation der Seite wurde relativ häufig benutzt, um die für eine Testperson interessanten Themengebiete anzusteuern.

6.6 Erkenntnisse auf Stackoverflow.com

Auf dieser Internetseite war zu beobachten, dass am häufigsten jene Links verwendet wurden, die zu Fragen navigieren. Dieser Bereich der Internetseite ist in Abbildung 4.5 türkis markiert.

Die Navigation der Seite mit Links wie beispielsweise „log in“ oder „careers“ wurde von keiner Testperson verwendet.

6.6.1 Verwendete Link-Texte

Ein Beispiel eines verwendeten Sprachbefehles ist: „show me the thread about java wrong image type“. Daraus lässt sich folgern, dass die Internetseite als Forum betrachtet wurde, das aus Themen (engl. „Threads“) besteht. Außerdem wurde der Link-Text wiederum auf die Wörter beschränkt, die das Thema beschreiben. Es wurde das Verb „returns“ entfernt und in diesem Fall das Pronomen „wrong“ verwendet, das den „Image Type“ beschreibt, der von der Programmiersprache Java zurückgeliefert wird.

6.6.2 Erkennung von Abkürzungen

Wie bereits in Abschnitt 6.2.2 erwähnt, beinhalten die Links auf der Seite sehr viele Abkürzungen. Der größte Kritikpunkt an der Funktionalität des Systems war die oft ungenaue Erkennung von Abkürzungen. Dieses Problem ist jedoch meist auf das Spracherkennungssystem zurückzuführen, da Abkürzungen bzw. deren ausgesprochene Varianten häufig nicht korrekt erkannt werden. Das Problem ließ sich meist durch eine langsamere und klarere Aussprache der Abkürzung vermindern.

6.6.3 Verwendete Schlüsselwörter

Wie bereits beschrieben, wurde von einer Testperson die Internetseite als Forum wahrgenommen und die Sprachbefehle daher mit den entsprechenden englischsprachigen Schlüsselwörtern gebildet.

Ansonsten wurden außer jenen zur Navigation sehr wenige Schlüsselwörter verwendet. Ausnahmen war hierbei eben die Testperson, welche die Seite als Forum verwendete.

6.7 Zusammenfassung

Die praktischen Versuche ergaben, dass das System für eine einfache Steuerung von Internetseiten ausreicht. Insbesondere dann, wenn die hauptsächlich verwendeten Elemente Navigations-Links sind.

Die Erkennung von längeren Sätzen war besser als jene von Abkürzungen, was die Steuerung von Stackoverflow etwas beeinträchtige.

Vergleicht man die Verwendung der Navigation der beiden Internetseiten, kann daraus abgeleitet werden, dass die Navigation auf Stackoverflow.com eine untergeordnete Rolle spielt, während sie auf einer Informationsseite wie BBC.com häufiger verwendet wird.

Kapitel 7

Schlussfolgerungen, Fragen und Ausblick

7.1 Allgemeine Schlussfolgerungen

Grundsätzlich kann festgehalten werden, dass eine automatische Generierung von Sprachkommandos zur Steuerung von HTML-Seiten nur bedingt möglich ist. Es ist zwar sehr einfach möglich, Textwerte aus HTML-Seiten zu extrahieren, diese entsprechen jedoch in den seltensten Fällen einem grammatikalisch korrekten Sprachbefehl.

Somit ist es praktisch unmöglich eine exakte Sprachgrammatik für ein Spracherkennungssystem zu definieren, ohne eine wissensbasierte Darstellung der Informationen der Internetseite zu erstellen. Durch die Verwendung einer Volltextsuche wie Lucene reicht jedoch eine einfache schriftliche Repräsentation eines HTML-Elements aus, um dieses zu identifizieren.

Problematisch wird dieser Ansatz immer dann, wenn die grundlegenden Richtlinien zur Barrierefreiheit im Internet auf einer Seite nicht angewendet werden. Dies würde bedeuten, dass

- Elemente keine oder nicht korrekte alternative schriftliche Repräsentationen besitzen,
- Link-Texte das Ziel eines Links nicht ausreichend beschreiben
- und Plug-Ins oder Bilder anstatt von Markup verwendet wird.

Aufgrund der Ergebnisse der praktischen Tests konnte die Verwendung der Schlüsselwörter nicht vollständig belegt werden. Diese wurden generell nur selten eingesetzt und beschränkten sich auf Navigationswörter wie beispielsweise „go to“ oder „navigate to“.

Ein direkter Zusammenhang der Schlüsselwörter und einem Bereich der Internetseite bzw. dem Ziel eines Links konnte nicht bestätigt werden. Ausnahmen hierbei ware je eine der Testpersonen, die zur Bildung der Sprachbefehle sowohl auf der Seite der BBC als auch auf Stackoverflow die Links zu

Artikeln bzw. Fragen Schlüsselwörter wie „article“ oder „question“ verwendete.

Diese Ergebnisse sind jedoch nicht zwangsläufig korrekt, da die Untersuchung keine statistisch belegte Relevanz besitzt und sie keiner „User Study“ entspricht. Unabhängig davon lassen sich jedoch Tendenzen erkennen.

7.2 Offene Fragen

Einige der aufgestellten Theorien konnten sowohl aufgrund von technischen Einschränkungen als auch aufgrund von zu großem Umfang nicht weiter untersucht werden. Diese werden in diesem Abschnitt erläutert.

7.2.1 Klassifikation der Internetseiten

Die Klassifikation von Internetseiten wurde an mehreren Stellen der Arbeit erwähnt, in Abschnitt 4.7 wird die Thematik kurz beschrieben. Zu untersuchen wäre hierbei, ob sich die Theorie bestätigen lässt, dass das Thema einer Internetseite Einfluss auf die Sprachbefehle hat. Dazu müsste ein entsprechender Algorithmus in das System eingebaut und für alle Arten von Internetseiten entsprechende Schlüsselwörter definiert werden.

Die zu beantwortenden Fragen wären die folgenden:

- Welchen Einfluss hat die Art einer Internetseite auf die Sprachbefehle?
- Kann aus der Art der Seite berechnet werden, welche Schlüsselwörter zu verwenden sind?

7.2.2 Mikroformate

Mikroformate sind Möglichkeiten, HTML-Elemente über CSS-Klassen semantisch zu annotieren. Dazu werden Klassen definiert, die auf allen Internetseiten, die sie einsetzen das selbe bedeuten. Unter [28] werden Mikroformate für verschiedene Einsatzzwecke gesammelt und Informationen über die Entwicklung neuer Formate gegeben.

Zu beantworten wären hierbei die folgenden Fragen:

- Existiert bereits ein Mikroformat, dass für die Zwecke der Generierung von Sprachbefehlen verwendet werden kann?
- Ist es sinnvoll ein eigenes Mikroformat zu erstellen?
- Kann die Generierung der Sprachbefehle merkbar verbessert werden?

7.3 Mögliche Weiterentwicklungen

Da das vorgestellte System nur einige Grundfunktionalitäten bietet, gibt es einige Möglichkeiten zur Weiterentwicklung. Diese werden im Folgenden kurz beschrieben.

7.3.1 Performance

Die Performance des Systems ist insbesondere dadurch, dass die Sprachbefehle bei jedem Aufruf einer Internetseite erneut berechnet werden, nicht gut. Die Wartezeit zum Laden einer neuen Seite wäre für einen Produktiveinsatz nicht geeignet.

Eine Möglichkeit zur Steigerung der Performance wäre das Vorausberechnen der Sprachbefehle. Dies könnte entweder einmal täglich oder bei jeder Änderung der Seite erfolgen. Beim Vorausberechnen würde die Problematik bestehen, dass neue Inhalte nicht mit Sprachbefehlen steuerbar wären, da für sie noch keine Befehle berechnet wurden. Aus diesem Grund wäre diese Möglichkeit nur für Internetseiten denkbar, deren Inhalt sich nicht während eines Tages verändert.

Die zweite Möglichkeit würde bedeuten, dass beim Eintragen eines Artikels oder beim Stellen einer Frage die Befehle berechnet werden müssten. Würde man die beiden Möglichkeiten kombinieren, könnte man täglich die aktuellen Sprachbefehle berechnen und während des Tages nur Befehle für die Änderungen der Seite.

7.3.2 Klassifizierung

Wie bereits einige Male erwähnt, könnte die Klassifizierung einer Internetseite dabei helfen, die geeigneten Schlüsselwörter auszuwählen und die Sprachbefehle anzupassen. Dies setzt außerdem voraus, dass es für jede Klasse von Internetseiten einen Satz von Befehlen und Schlüsselwörtern gibt, die entsprechend eingesetzt werden.

7.3.3 Navigation über mehrere Seiten

Die aktuelle Systemarchitektur setzt voraus, dass ein Benutzer nur eine einzige Seite bedienen kann. Interessant wäre, eine Navigation über mehrere Seiten hinweg zu ermöglichen. Dies würde von Seiten des Benutzers ein Wissen über die zuerst angewählte Seite voraussetzen.

Zu diesem Zweck wäre eine Komponente im System nötig, die einen Sprachbefehl in dessen Bestandteile aufteilt und die daraus entstehenden Befehle auf den jeweiligen Seiten ausführt. Besser wäre hierbei eine Möglichkeit, trotz einer Navigation über mehrere Seiten hinweg, nur die zuletzt angesteuerte an den Client zu liefern.

Aufgrund der Aneinanderreihung von Spracherkennungs- und Suchtoleranzen könnte jedoch der resultierende Fehler sehr groß werden und eine zielgerichtete Verwendung der Internetseite verhindern.

Kapitel 8

Zusammenfassung

In der vorliegenden Arbeit wurde erarbeitet, wie auf Basis der HTML-Spezifikation und einigen Richtlinien zur Barrierefreiheit die Berechnung von Sprachbefehlen durchgeführt werden kann.

Es wurden die Möglichkeiten zur Interaktion mit einer Internetseite aufgezeigt und die dazu benötigten HTML-Elemente beschrieben. Aus diesen Elementen wurden schriftliche Repräsentationen berechnet, welche als Basis für Sprachbefehle verwendet werden.

Diese schriftlichen Werte sind keine grammatikalisch korrekten Sätze, sondern meist nur Teile eines Satzes oder beinhalten nur Stichwörter und Abkürzungen. Daher wurden die Sprachbefehle nicht in einem starren und fehlerintoleranten Speicher abgelegt sondern mit der Suchlösung „Apache Lucene“ indiziert.

Die Sprachbefehle werden mit Schlüsselwörtern angereichert, die dem Kontext des Elementes innerhalb der Seite entsprechen. Dazu wurden einige Möglichkeiten betrachtet, um diesen Kontext zu ermitteln.

Einige der aufgestellten Theorien wurden anschließend in einem praktischen Versuch mit einem System überprüft. Die Beschreibung der Architektur und der Funktionsweise des Systems sind in einem eigenen Kapitel enthalten.

Durch die praktischen Tests konnte festgestellt werden, dass die meisten Testpersonen die Schlüsselwörter nicht an den Kontext des Elementes anpassen, wohl aber an die Art der Internetseite.

Abbildungsverzeichnis

2.1 Einflüsse auf ein Sprachsignal 4
2.2 Informationstheoretische Sicht der Spracherkennung 6
2.3 Ablaufdiagramm der Grammatik aus Listing 2.1 8
2.4 Funktionsweise von Apache Lucene 19

4.1 Beispiel für einen Blog-Artikel mit „Read more“-Link 31
4.2 Einige Euro-Münzen 32
4.3 Startseite von BBC.com vom 19. April 2012 47
4.4 Sport-Kategorie der Seite http://www.bbc.com vom 8. Juni 2012 49
4.5 Startseite von http://www.stackoverflow.com vom 30. März 2012 52

5.1 Grundlegende Systemarchitektur des Masterprojektes 56

6.1 Verwendung der gesamten Seitenhöhe als Scroll-Weite 72

Tabellenverzeichnis

2.1 Widget-Rollen aus WAI-ARIA [30, Abschnitt 5.3.2] 16
2.2 „Landmark“ Rollen aus WAI-ARIA [30, Abschnitt 5.3.4] . . . 16
2.3 Beispiele für WAI-ARIA-Zustände 17
2.4 Beispiele für WAI-ARIA-Relationen 17
2.5 HTML4 Element-Gruppen [32, Abschnitt 21] 23
2.6 Relevante neue Elemente in HTML5 24

4.1 Umlaute und deren Ersatzzeichen 35
4.2 Sonderzeichen und deren Ersatzworte 35
4.3 Beispiele für Zahlen der Google Spracherkennung 36
4.4 Bruchzahlen und deren Ersatz-Worte 37

5.1 Maus-Events der HTML-Spezifikation 59
5.2 Tastatur-Events der HTML-Spezifikation 60

6.1 Zum Scrollen auf einer Seite verwendete Sprachbefehle 71

Anhang A

Installationsanleitung des Masterprojektes

A.1 Voraussetzungen

Diese Installationsanleitung geht davon aus, dass bereits ein funktionierendes Linux-System für den Betrieb des Web-Servers zur Verfügung steht. Wenn dies nicht der Fall ist, steht unter [37] eine Installationsanleitung zur Verfügung. Die Installation muss nicht zwingend auf einem physischen Rechner, sondern kann auch innerhalb einer virtuellen Maschine erfolgen.

A.2 Apache 2 HTTPD Server

Als Basis dient hierbei das Standardpaket, das mit der jeweiligen Linux-Distribution geliefert wird. Entwickelt und getestet wurde die Anwendung unter Ubuntu 10.04.3 64Bit mit Apache 2.2.16.

Die Installation kann mit zwei unterschiedlichen Methoden durchgeführt werden. Die einfachste Möglichkeit bietet die Installation mit Tasksel, die in Abschnitt A.2.1 erläutert wird.

Alle Befehle müssen in ein Terminal-Fenster eingegeben werden.

A.2.1 Installation mit Tasksel

Tasksel bietet eine Möglichkeit, einen fertigen Web-Server inklusive Apache2, PHP sowie einem MySQL-Server zu installieren. Diese Grundinstallation wird auch als LAMP (Linux, Apache, MySQL, PHP) bezeichnet.

Dies hat allerdings in diesem Fall den Nachteil, dass ein MySQL-Server installiert wird, der zur Ausführung des Projektes nicht benötigt wird.

Tasksel installieren

Zur Installation muss folgender Code ausgeführt werden:

```
1 sudo apt-get install tasksel
```

LAMP Server installieren

Nun kann mit Tasksel der Server installiert werden:

```
1 sudo tasksel install lamp-server
```

A.2.2 Manuelle Installation

Zur manuellen Installation müssen die benötigten Pakete einzeln installiert werden. Dies kann über den folgenden Befehl erfolgen:

```
1 sudo apt-get install apache2 libapache2-mod-php5 php5-common
```

A.2.3 Rewrite-Modul konfigurieren

Das Rewrite-Modul kann dazu verwendet werden, Anfragen an eine bestimmte Ressource des Apache Servers intern auf eine andere Ressource umzuleiten. Dies geschieht anhand von sogenannten „Rewrite-Regeln“, die mithilfe von regulären Ausdrücken diese Umleitungen definieren.

Das Rewrite-Modul ist im Normalfall bereits installiert und muss nur aktiviert werden. Dies geschieht mit dem folgenden Befehl:

```
1 sudo a2enmod rewrite
```

Nun muss noch die benötigte Rewrite-Regel definiert werden, die alle Anfragen auf eine HTML-Seite intern auf das PHP-Skript umleitet, das die weitere Verarbeitung vornimmt. Für den „Default VirtualHost“ muss die folgende Regel definiert werden:

```
1  <Directory /var/www/>
2    RewriteEngine on
3    RewriteBase /
4    RewriteRule ^(.*).html$ inject.php?page=$1
5
6    Options Indexes FollowSymLinks MultiViews
7    AllowOverride None
8    Order allow,deny
9    allow from all
10 </Directory>
```

A.3 Benötigte Skripte und Dateien installieren

Die folgenden Dateien müssen sich im Document-Root des Server befinden:

- DOMDocumentGrammarGenerator.class.php
- Link.class.php
- checksession.inc.php

- controller.js
- domchange.php
- index.php
- inject.php
- jquery.js
- jquery-noconflict.js
- speechcontroller.php
- settings.inc.php

Diese Dateien befinden sich auf dem beiliegenden Datenträger.

A.3.1 Lucene

Für Lucene wird das Zend-Framework benötigt. Dieses befindet sich als ZIP-Datei auf dem Datenträger, der dieser Arbeit beiliegt. Das Framework muss am Server installiert und der entsprechende Ordner in den Include-Pfad von PHP eingefügt werden.

Dazu muss der Ordner „Zend“ in entpackter Form am Computer installiert werden. Beispielsweise unter „/usr/local/share/zend/Zend“.

In der „php.ini“[1] Datei muss nun der Include-Pfad angepasst werden. Die entsprechende Zeile befindet sich in etwa bei Zeile 785 und muss auskommentiert werden. Danach muss der Pfad zum Zend Framework hinzugefügt werden.

Vorher:

```
1 ; include_path = "`.:/usr/share/php"'
```

Nachher:

```
1 include_path = "`.:/usr/share/php:/usr/local/share/zend"'
```

Damit die Änderungen wirksam werden, muss der Apache Server neu gestartet werden. Dies geschieht mit dem folgenden Befehl:

```
1 sudo service apache2 restart
```

[1] Befindet sich auf Ubuntu-Systemen in „/etc/php5/apache2“

Quellenverzeichnis

Literatur

[1] Alexander Clark, Chris Fox und Shalom Lappin. *The Handbook of Computational Linguistics and Natural Language Processing*. 1. Auflage. Chichester, UK: Wiley-Blackwell, 2010.

[2] Michael H. Cohen, James P. Giangola und Jennifer Balogh. *Voice User Interface Design*. Amsterdam: Addison-Wesley Longman, 2004.

[3] Eric J. Glover u. a. „Using web structure for classifying and describing web pages“. In: *Proceedings of the 11th international conference on World Wide Web*. WWW '02. Honolulu, Hawaii, USA: ACM, 2002, S. 562–569. URL: http://doi.acm.org/10.1145/511446.511520.

[4] Alexander Gruenstein, Ian McGraw und Ibrahim Badr. „The WAMI toolkit for developing, deploying, and evaluating web-accessible multimodal interfaces“. In: *Proceedings of the 10th international conference on Multimodal interfaces*. ICMI '08. New York, NY, USA: ACM, 2008, S. 141–148. URL: http://doi.acm.org/10.1145/1452392.1452420.

[5] Simon Harper und Yeliz Yesilada. *Web Accessibility - A Foundation for Research*. Hrsg. von John Karat und Jean Vanderdonckt. 1. Auflage. London: Springer-Verlag London, 2008.

[6] Helmut Herold. *lex und yacc: Lexikalische und syntaktische Analyse*. 2. Auflage. Bonn; Paris: Addison-Wesley, 1995.

[7] Min-Yen Kan und Hoang Oanh Nguyen Thi. „Fast webpage classification using URL features“. In: *Proceedings of the 14th ACM international conference on Information and knowledge management*. CIKM '05. Bremen, Germany: ACM, 2005, S. 325–326. URL: http://doi.acm.org/10.1145/1099554.1099649.

[8] Bruce Lawson und Remy Sharp. *Introducing HTML5*. 2. Auflage. Berkeley, CA, USA: New Riders Press, 2011.

[9] Michael McCandless, Erik Hatcher und Otis Gospodnetic. *Lucene in Action*. 2. Auflage. Greenwich, CT, USA: Manning Publications Co., 2010.

[10] Beat Pfister und Tobias Kaufmann. *Sprachverarbeitung: Grundlagen und Methoden der Sprachsynthese und Spracherkennung*. 1. Auflage. Berlin Heidelberg: Springer-Verlag, 2008.

[11] Willie Walker u. a. *Sphinx-4: A Flexible Open Source Framework for Speech Recognition*. 2004. URL: http://cmusphinx.sourceforge.net/sphinx4/doc/Sphinx4Whitepaper.pdf.

Online-Quellen

[12] *Allan Jardine | Reflections | Visual Event 2*. URL: http://www.sprymedia.co.uk/article/Visual+Event+2 (besucht am 30.05.2012).

[13] *Applets*. 2012. URL: http://java.sun.com/applets/ (besucht am 21.03.2012).

[14] *Are image file names important to SEO?* 2. Juli 2009. URL: http://stackoverflow.com/questions/1072238/are-image-file-names-important-to-seo (besucht am 30.05.2012).

[15] Adam Bergkvist u. a. *WebRTC 1.0: Real-time Communication Between Browsers*. 2011. URL: http://dev.w3.org/2011/webrtc/editor/webrtc.html (besucht am 17.03.2012).

[16] Daniel C. Burnett und Anant Narayanan. *getusermedia: Getting access to local devices that can generate multimedia streams*. 2011. URL: http://dev.w3.org/2011/webrtc/editor/getusermedia.html (besucht am 17.03.2012).

[17] Ben Caldwell u. a. *Web Content Accessibility Guidelines (WCAG) 2.0*. 2008. URL: http://www.w3.org/TR/2008/REC-WCAG20-20081211/ (besucht am 05.04.2012).

[18] Wendy Chisholm, Gregg Vanderheiden und Ian Jacobs. *Web Content Accessibility Guidelines 1.0*. 1999. URL: http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505/ (besucht am 05.04.2012).

[19] Darcy Clarke. *Detect Attribute Changes with jQuery*. 2010. URL: http://darcyclarke.me/development/detect-attribute-changes-with-jquery/ (besucht am 03.05.2012).

[20] *Extreme nesting experiment in IE and Firefox*. 2007. URL: http://helephant.com/2007/08/04/extreme-nesting-experiment-in-ie-and-firefox/ (besucht am 15.05.2012).

[21] Google. *Speech Input API Specification*. 2010. URL: http://lists.w3.org/Archives/Public/public-xg-htmlspeech/2011Feb/att-0020/api-draft.html (besucht am 05.04.2012).

[22] Google. *Testing WebRTC on Chrome*. 2012. URL: http://www.webrtc.org/running-the-demos (besucht am 05.04.2012).

[23] Alexander Gruenstein und Ian McGraw. *WAMI News*. 2012. URL: http://wami.csail.mit.edu/news.php (besucht am 17.03.2012).

[24] Andrew Hunt. *JSpeech Grammar Format*. 2000. URL: http://www.w3.org/TR/2000/NOTE-jsgf-20000605/ (besucht am 21.05.2012).

[25] *Image SEO - 6 Optimization Tips for Images*. 11. Juni 2010. URL: http://www.toprankblog.com/2010/06/6-tips-image-seo/ (besucht am 30.05.2012).

[26] Anne van Kesteren und Simon Pieters. *HTML5 differences from HTML4*. URL: http://www.w3.org/TR/2012/WD-html5-diff-20120329/ (besucht am 30.05.2012).

[27] *Keynote - The differences between WCAG1 and WCAG2 - Gian Wild*. 2012. URL: http://www.youtube.com/watch?v=mGbefqDDspM.

[28] Microformats.org. *Microformats | Get Started*. URL: http://microformats.org/get-started (besucht am 01.06.2012).

[29] MySQL. *MySQL :: MySQL 5.1 Referenzhandbuch :: 12.7 MySQL-Volltextsuche*. 2012. URL: http://dev.mysql.com/doc/refman/5.1/de/fulltext-search.html (besucht am 26.04.2012).

[30] Lisa Pappas, Richard Schwerdtfeger und Lisa Seeman. *Accessible Rich Internet Applications (WAI-ARIA) 1.0*. 2011. URL: http://www.w3.org/TR/2011/CR-wai-aria-20110118/ (besucht am 22.05.2012).

[31] Axel Postinett. *Spracherkennung: So will Google zum Sprachgenie werden*. 2011. URL: http://www.handelsblatt.com/technologie/forschung-medizin/forschung-innovation/spracherkennung-die-sprachscouts-gingen-sogar-auf-tauchstation/5798652-2.html (besucht am 15.03.2012).

[32] Dave Raggett, Arnaud Le Hors und Ian Jacobs. *HTML 4.01 Specification*. 1999. URL: http://www.w3.org/TR/1999/REC-html401-19991224 (besucht am 05.04.2012).

[33] Dottoro Web Reference. *DOMAttrModified Event JavaScript*. URL: http://help.dottoro.com/ljdchxcl.php (besucht am 03.05.2012).

[34] Apache Solr. *Apache Lucene - Apache Solr*. 2012. URL: http://lucene.apache.org/solr/ (besucht am 27.04.2012).

[35] SpeechAPI.com. *Javascript Api*. 2012. URL: http://www.speechapi.com/apis/advancedjs/ (besucht am 21.03.2012).

[36] *Stackoverflow.com - What is the maximum depth of HTML documents in practice*. 2011. URL: http://stackoverflow.com/questions/7770573/what-is-the-maximum-depth-of-html-documents-in-practice (besucht am 15.05.2012).

[37] Ubuntuusers.de. *Ubuntu Installation*. 2012. URL: http://wiki.ubuntuusers.de/Ubuntu_Installation (besucht am 14.03.2012).

[38] W3C. *Links in HTML documents*. URL: http://www.w3.org/TR/html401/struct/links.html (besucht am 15.05.2012).

[39] W3C. *WAI-ARIA Taxonomy*. 2011. URL: http://www.w3.org/TR/2011/CR-wai-aria-20110118/rdf_model (besucht am 19.04.2012).

[40] Wikipedia. *Barrierefreiheit - Wikipedia*. 2012. URL: http://de.wikipedia.org/w/index.php?title=Barrierefreiheit\&oldid=100405935 (besucht am 21.03.2012).

[41] Wikipedia. *Braillezeile — Wikipedia*. 2012. URL: http://de.wikipedia.org/w/index.php?title=Braillezeile&oldid=91594126 (besucht am 17.03.2012).

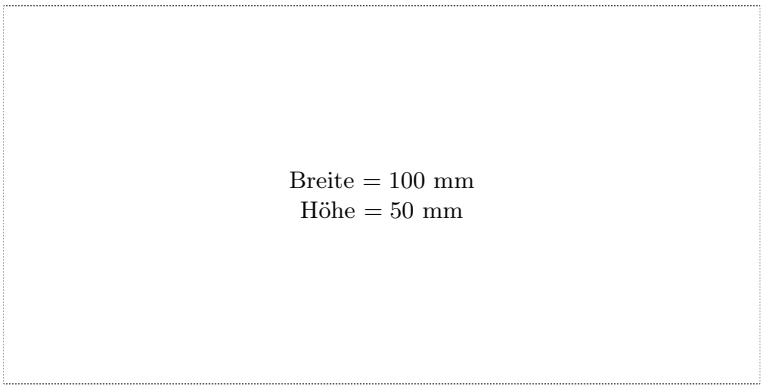
[42] Wikipedia. *Hadoop - Wikipedia*. 2012. URL: http://de.wikipedia.org/w/index.php?title=Hadoop&oldid=101610235 (besucht am 21.03.2012).

[43] Wikipedia. *Regulärer Ausdruck — Wikipedia*. 2012. URL: http://de.wikipedia.org/w/index.php?title=Regul%C3%A4rer_Ausdruck&oldid=104117275 (besucht am 21.05.2012).

[44] Zend. *Zend Framework: Documentation: Overview - Zend Framework Manual*. 2012. URL: http://framework.zend.com/manual/en/zend.search.lucene.overview.html (besucht am 26.04.2012).

Messbox zur Druckkontrolle

— Druckgröße kontrollieren! —



— Diese Seite nach dem Druck entfernen! —