UNIVERSITÄT DORTMUND
FACHBEREICH INFORMATIK

Diplomarbeit

URLChecker – ein Agent zur intelligenten, seiteninhaltsbasierten URL-Überprüfung

André Masloch

Diplomarbeit am
Fachbereich Informatik
der Universität Dortmund

18. November 2003

Betreuer:
Prof. Dr. Katharina Morik
Dipl.-Inform. Stefan Haustein

Für Patricia

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Bei jedem Benutzer der im WWW surft wird sich nach und nach eine Linksammlung (Bookmarks) mit Verweisen auf interessante Seiten bilden, die im Laufe der Zeit immer größer wird. Nun ist das WWW ein riesiges Netz, in dem sich die Informationen bzw. Inhalte sehr schnell ändern. Was heute noch unter dieser Adresse zu erreichen war, kann morgen schon unter einer anderen Adresse zu finden sein.

Für die Linksammlung des Benutzers heißt das, daß einige Links nach relativ kurzer Zeit nicht mehr den Inhalt bieten, wegen dem er diese Seite aufgenommen hatte. Es ist also eine regelmäßige Pflege der Linksammlung erforderlich. Im Normalfall ist diese per Hand vom Benutzer zu erledigen, indem alle URLs in regelmäßigen Zeitabständen kontrolliert werden.

Wäre es nicht besser wenn diese Aufgabe automatisch ausgeführt werden könnte? Oder wäre nicht auch schon ein Hinweis der Art „Hallo, die Seite `www.foo.com/bar.html` hat sich geändert“ hilfreich?

Ein weiterer Fall ist der Administrator eines Portals im WWW. Das Portal besteht aus vielen Verweisen auf andere Seiten, die sich mit der Zeit zunehmend ändern. Also ist auch hier eine regelmäßige Pflege unabdingbar. Aufgrund der Anzahl der Links im Portal ist die manuelle Pflege per Hand aber nicht mehr ausführbar – der Administrator wäre den gesamten Tag nur noch damit beschäftigt, Seiten zu überprüfen.
Außerdem gibt es Agenten-Systeme, die unter Umständen sehr viele URLs überwachen. Oft findet sich unter einer angegebenen URL gar kein Eintrag mehr, oder nur ein Hinweis auf eine neue URL, unter der die alte Information zu finden ist. Das kann von den gängigen Agenten-Systemen nicht verarbeitet werden, aber in diesen Systemen soll nicht ausgerechnet für solche Überprüfungen ein Mensch eingesetzt werden.

1.1 Aufgabenstellung

Ziel dieser Diplomarbeit ist es, einen Agenten zu erstellen, der eine gegebene Menge von `URL`s auf ihren Inhalt prüft. Wenn sich der Inhalt im Laufe der Zeit ändert oder verschiebt, kann der Benutzer dies anhand des Status der `URL` im `URLChecker` erkennen. Der Agent ist dabei lernfähig, d.h. er kann mit Beispielen trainiert werden, anhand derer er seine späteren Bewertungen anpasst.

Nach der Inbetriebnahme des `URLChecker`s ist zunächst eine Anpassung des verwendeten Modells nötig, da der Benutzer die Dokumente auf unterschiedliche Weise gebrauchen kann (siehe 2.1). Durch den Einsatz vieler verschiedener Distanzmaße (siehe Kapitel 4.5) versucht der Agent sich an die unterschiedlichen Präferenzen des Benutzers anzupassen.

Der in dieser Diplomarbeit entwickelte Agent ist derzeit auf die Verarbeitung von HTML-Texten beschränkt. Grafiken, JAVA-Scripts und andere Formate werden nicht berücksichtigt. Da der Agent die Klassenbibliothek `BotIShelly` ([Banken et al., 2000]) benutzt, ist eine Erweiterung mit anderen Formaten nicht ausgeschlossen.

Im Rahmen der Arbeit sollen die folgenden Fragen beantwortet werden:

- Ist die Verwendung von Seitenpaaren als Grundlage geeignet, um aus Seitenbewertungen zu lernen?
- Ist der Einsatz mehrerer Distanzmaße in Kombination sinnvoll?
- Führt die Kombination mehrerer Distanzmaße immer zu einer Verbesserung der Ergebnisse?

1.2 Aufbau der Diplomarbeit

In Kapitel 2 werden nach einem kurzen Überblick zunächst die unterschiedlichen Gebrauchsarten und der Grundaufbau von `URL`s behandelt. Nachdem die unterschiedlichen Fehlerarten und die damit verbundenen Schwierigkeiten erklärt wurden, wird in Kapitel 3 ein Vergleich zu anderen Alternativen zur Lösung des Problems gezogen. Anschließend folgt eine Auswahl von Techniken, die in der Realisierung (Kapitel 4) zur Implementierung des `URLChecker`s genutzt werden.

Nachdem in Kapitel 4 zunächst ein Überblick über die Architektur und Vorgehensweise des Agenten geschafft wurde, werden in Kapitel 4.5 die im Agenten verwendeten Distanzmaße ausführlich beschrieben.

In Kapitel 5 folgt dann die Beschreibung und Auswertung der verschiedenen Testfälle. Diese sind die Grundlage für die anschließende Evaluation des Agenten.

Kapitel 6 fasst die grundlegenden Ergebnisse aus den vorhergehenden Kapiteln zusammen und gibt einen Ausblick über weitere Möglichkeiten und Verbesserungen des Agenten und seiner verwendeten Techniken.

Kapitel 2

Grundlagen

Ziel diese Kapitels ist es, die grundlegenden Konzepte zu vermitteln, die zum Verständnis der später benutzten Techniken und Distanzmaße benötigt werden. Zunächst wird auf die Bedeutung der `URL` und des Inhalts eingegangen. Anschließend wird der Aufbau der `URL` erläutert, der bei den in Kapitel 4.5 beschriebenen Distanzmaßen zur Bewertung der Ähnlichkeit von Dokumenten benutzt wird. In Kapitel 2.3 werden dann die unterschiedlichen Probleme erläutert die auftreten können, wenn eine Seite betrachtet bzw. verglichen werden soll. Es ist zum Beispiel ein großer Unterschied, ob ein Server nicht erreichbar ist und deshalb der Inhalt der Seite nicht verfügbar ist, oder ob sich die Seite tatsächlich im Inhalt geändert hat. Im übrigen besitzen die verschiedenen Fehler unterschiedliche Schwierigkeitsgrade, d.h. einige Fehler sind für einen Agenten leichter zu erkennen als andere. Gerade die Inhaltsänderung ist ein schwieriges Problem. Wann hat sich die Seite *geändert*? Sind kleine Änderungen noch zu tolerieren? Aber hier stellt sich leider schon die nächste Frage: Was sind *kleine* Änderungen?

Ein weiteres Problem stellt sich in der Art und Weise, wie eine `URL`, ein Link oder ein Dokument vom Benutzer angesehen wird (siehe Kapitel 2.1). Ist der Benutzer am konkreten Inhalt der Seite interessiert, oder nicht?
Dieses Problem läßt sich gut am Beispiel einer Seite mit dem aktuellen Kinoprogramm zeigen. Auf der einen Seite kann der Benutzer *genau* diese Seite haben wollen, andererseits kann es aber auch sein, das der Benutzer nur das aktuelle Programm sehen möchte. Im ersten Falle wären spätere Versionen der Seite anders als die derzeit aktuelle Version der Seite anzusehen, im zweiten Falle wäre dies genau umgekehrt. Aber wie soll ein Agent entscheiden, welche Gebrauchsart der Benutzer verwendet, zumal es dem Benutzer meist selbst nicht bewußt ist? Die Definition des „Seiteninhaltes“ hängt also auch von der Gebrauchsart ab. Noch ein Problem mit dem der Agent umgehen muß.

All diesen Problemen, die im folgenden noch genauer behandelt werden, steht der in dieser Arbeit entwickelte Agent (im folgenden `URLChecker` genannt) gegenüber. Der Agent soll den Inhalt der überwachten Seiten zu unterschiedlichen Zeitpunkten überprüfen, und

gegebenenfalls anhand des neuen Inhaltes entscheiden, ob dieser noch ähnlich zum alten Inhalt ist. Der Benutzer soll dabei nicht notwendigerweise anwesend sein, sondern nur später das Ergebnis sehen können, der `URLChecker` ist also ein Offline-Agent. Revidiert der Benutzer eine Entscheidung des `URLChecker`s, so soll es möglich sein das sich der Agent diesen Korrekturen anpasst.

2.1 Gebrauchsarten

Wie schon in der Einleitung des Kapitels verdeutlicht, gibt es unterschiedliche Weisen, wie der Benutzer ein Dokument oder eine `URL` benutzen kann. Diese Weisen werden in der Linguistik als Gebrauchsarten bezeichnet. Zur Erklärung des *referentiellen* und *attributiven* hier ein Beispiel:

> Der Mörder von Schmidt muß wahnsinnig sein

1. Referentieller Gebrauch
   Wir kennen den Mörder von Schmidt. Während seines Prozesses verhält sich der Mörder (Katzenbach) sehr auffällig. Dann kann man den Satz dazu verwenden, um zu sagen, daß Katzenbach unsererer Meinung nach wahnsinnig ist.

2. Attributiver Gebrauch
   Wir kennen den Mörder von Schmidt nicht, sehen aber das das Opfer auf bestialische Weise getötet wurde. Dann kann der obige Satz dazu benutzt werden um auszudrücken: „Wer auch immer Schmidt ermordet hat, der muß wahnsinnig sein“.

Im ersten Fall ändert sich der Bezug auf Katzenbach selbst dann nicht, wenn sich später herausstellen sollte, daß Katzenbach doch nicht der Mörder ist, im zweiten Fall ist dies sehr wohl der Fall.
Ein weiteres Beispiel ist „Der Bundeskanzler von Deutschland“. Ist nun der Bundeskanzler als Person, oder als Träger des Amtes gemeint? Im ersten Falle würde sich die Person nicht ändern, auch wenn später jemand anderes Bundeskanzler ist, im zweiten Fall schon.

Wenn man die Gebrauchsarten auf Dokumente und `URL`s (z.B. Links) überträgt, so ergeben sich auch für diese unterschiedliche Arten des Gebrauchs. Diese werden in den folgenden Unterkapiteln jeweils für die einzelnen Dokument-Teile behandelt.

2.2 Das Dokument

Die grundlegende „Einheit“ für den URLChecker ist das Dokument, welches sich aus zwei Teilen zusammensetzt:

1. Dokument-URL
   Die Dokument-URL beschreibt den Ort des Dokumentes im WWW. Über die URL kann der Benutzer auf den Inhalt des Dokumentes zugreifen, man könnte die URL also als eine Art Referenz auf den Inhalt auffassen. Oft sind in der URL schon Informationen über den Inhalt vorhanden, so handelt das Dokument unter http://www-ai.cs.uni-dortmund.de/LEHRE/DIPLOM/OFFEN über noch offene Diplomarbeiten am Lehrstuhl 8 der Universität Dortmund, es gibt aber auch viele Dokumente, bei denen dies nicht so ist. Bei einem datenbankgestützten WWW-Server kann man nur selten Schlußfolgerungen über den Inhalt aus der URL ziehen. So ist zum Beispiel nicht ersichtlich, das sich hinter http://freshmeat.net/articles/view/857/ ein Tutorial für die Konfiguration eines Open Source Mail Gateway's befindet.

   Es stellt sich die Frage, ob Informationen aus der URL dazu verwendet werden können, Unterschiede zwischen Dokumenten zu erkennen oder nicht. Weitere Problematiken im Bezug auf Dokument-URLs werden in 2.2.1 aufgezeigt.

2. Dokument-Inhalt
   Mit dem Dokument-Inhalt sind primär die Daten gemeint die man erhält, wenn man eine URL von einem Server anfordert. Dabei wird im allgemeinen noch keine Annahme über das Format der Daten gemacht, d.h. der „Inhalt“ der dem Benutzer präsentiert wird muß dabei nicht mit dem Inhalt der URL übereinstimmen.

   Da der Agent zunächst für den Einsatz auf HTML-Seiten konzipiert ist, wird in Kapitel 2.2.2 näher auf spezielle Eigenschaften von HTML-Dokumenten eingegangen. Interessant für die Inhaltsanalyse können zum Beispiel die Links in einem Dokument sein. Es ist aber auch möglich, HTML-Tabellen zu erkennen. Mehr dazu ist in [Wang, Hu, 2002] zu finden.

2.2.1 Die Dokument-Adresse (URL)

Die URL (Uniform Resource Locator) repräsentiert einen Zeiger auf „Inhalt“ im WWW. Der „Inhalt“ kann hierbei eine Datei sein, aber zum Beispiel auch das Ergebnis auf eine Anfrage an eine Datenbank. Im Vergleich zur URI bezieht sich die URL auf den „Ort“ der Ressource im Netz, wohingegen sich die URI auch auf einen Namen oder sonstige Attribute der Ressource beziehen kann. Weitere Informationen über URL-Typen und Beispiele für Einsteiger finden sich unter [The NCSA Mosaic Team, 1998].

Wie in [Berners-Lee, 1996] beschrieben stellen sich bei der Verwendung von URIs einige Fragen zu deren Aussagekraft und Bedeutung. In verschiedenen Diskussionen wurde versucht, den Unterschied zwischen „Adresse“ und „Name“ aufzuzeigen. Der „Name“ wird im allgemeinen als fest zugehörig zu einem Objekt gesehen, die „Adresse“ als fest zugehörig zu einem Ort. Während der Lebenszeit eines Objektes ändert sich der „Name“ nicht, die „Adresse“ kann sich aber oft ändern.

Im Falle von Servernamen und IP-Adressen kommt man schlußendlich zum Ergebnis, das die IP-Adresse der „Adresse“ entspricht, und der Servername dem „Namen“, allerdings hat sich herausgestellt das diese Annahmen im wirklichen Leben immer hinterfragt werden müssen. Die beiden folgenden Beispiele machen klar, daß der „Name“ eines Objektes im WWW während seiner Lebenszeit nicht konstant ist, sondern wie die Adresse veränderlich:

- Die Adresse als feste Eigenschaft
  In einer Universität gab es einen Server auf den über seine IP-Adresse zugegriffen werden sollte. Grund: Der Servername war abhängig von der Abteilung, die den Server verwaltet und wegen einer Umstrukturierung in der Universität war diese momentan nicht bekannt. Die Verwaltung versicherte aber, daß die Adresse des Servers für eine lange Zeit gleich bleiben würde.

- `info.cern.ch`
  Nachdem die Koordination der WWW-Protokolle von CERN an MIT/LCS[1] abgegeben wurde, mußte der Name des Servers geändert werden, da sich die IP-Adresse des Servers durch die Abgabe geändert hatte[2]. Also wurden alle Links im Bereich von `info.cern.ch` geändert, so daß sie auf den Bereich `w3.org` zeigten.

2.2.1.1 Aufbau

Der Aufbau der URI[3] ist in [Berners-Lee et al., 1998] genau festgelegt. Für den Gebrauch in dieser Arbeit reicht jedoch eine vereinfachte Definition der Komponenten einer URL aus. Die einzelnen Komponenten werden in den Distanzmaßen verwendet, die zum Vergleich von Dokumenten genutzt werden (Kapitel 4.5).

1. Schema
   Durch das Schema wird der Aufbau aller folgenden Komponenten bestimmt. So hat zum Beispiel eine URI mit dem Schema `mailto:` andere Komponenten als eine

[1] Organisation des World Wide Web Consortiums (W3C)
[2] CERN hat eine Regel, die besagt das kein Name in der Domain `cern.ch` auf IP-Adressen zeigen durfte, die nicht physikalisch der CERN-Seite angehörten.
[3] Da die URL eine Untermenge der URI ist auch der Aufbau der URL durch den Aufbau der URI abgedeckt.

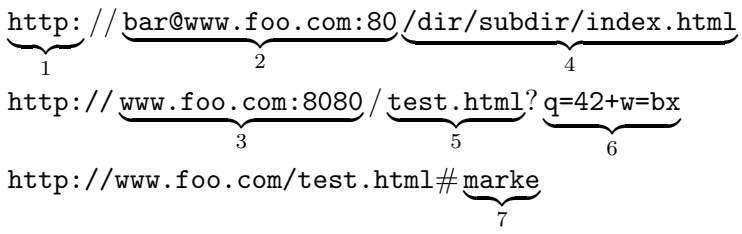


Abbildung 2.1: Schematischer Aufbau einer URI

`URI` mit dem Schema `http:`. Für den `URLChecker` sind nur `URI`s mit dem Schema `http:` relevant, der weitere Aufbau der `URI` bezieht sich deshalb auf dieses Schema.

2. Authority
   Die Authority-Komponente setzt sich aus Hostnamen, Portnummer sowie dem Benutzernamen zusammen. Der Benutzername wird gegebenenfalls zur Authentifizierung am Server benutzt.

3. Host
   Die Host-Komponente bezeichnet den Servernamen, auf dem das Dokument liegt. Der Bezeichner HOST(D, w)ird in späteren Kapiteln benutzt, um auf diese Komponente der `URL` zuzugreifen.

4. Path
   Die Path-Komponente identifiziert den Ort der Datei innerhalb des Servers. In den Anfängen war dies vergleichbar mit dem Pfad der Datei im Dateisystem auf dem Server. Heutzutage ist die Path-Komponente zwar noch oft auf einen Teil des Pfades abbildbar, im allgemeinen trifft dies aber nicht mehr zu.

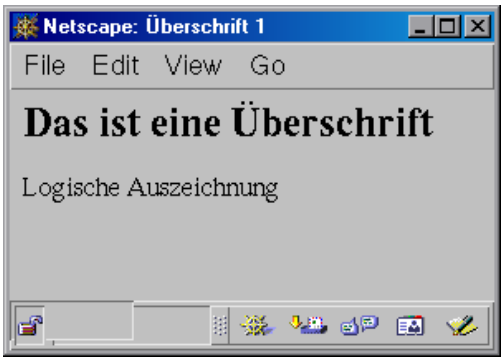
5. Filename
   Die Filename-Komponente besteht aus dem letzten Teil der Path-Komponente und bezeichnet im Regelfall den Dateinamen der Ressource. In späteren Kapiteln wird FILE(D_1, D_2) benutzt, um die Filename-Komponenten zweier Dokumente zu vergleichen.

6. Query
   Die Query-Komponente ist ein Informationsträger für Parameter o.ä., die von dem Server bzw. der Ressource interpretiert werden sollen. Im Zusammenhang mit serverbasierten Anwendungen (wie z.B. Servlets) wird diese Komponente neben dem Einsatz von HTML-Formularen oft genutzt, um Benutzerinformationen an den Server zu senden. Mit QUERY(D) kann auf diese Komponente zugegriffen werden.

7. Fragment
   Die Fragment-Komponente ist eine zusätzliche Referenzinformation für den Benutzer. Im Falle von HTML-Seiten wird die Fragment-Komponente zum Beispiel dafür benutzt, einzelne Verweise in einem Dokument anzuspringen. Im obigen Fall würde der Browser also nach dem Laden der Seite automatisch zur Marke `marke` im Dokument springen.

2.2.2 Der Dokument-Inhalt (Text)

Bei dem Großteil der Seiten im `WWW` wird HTML als Seitenbeschreibungssprache benutzt, welches eine logische Auszeichnung von Inhaltselementen mit Hilfe sogenannter Tags ermöglicht. Als Beispiel sei hier das Tag `<H1>` genannt, das den Webbrowser dazu veranlasst, eine größere Schrift zur Anzeige zu benutzen. Für einen Agenten ist es mitunter schwierig, diese Struktur zu verarbeiten und somit kann es auch schwierig sein, Seitenänderungen dieser Art zu erkennen. Wiederum dient das Tag `<H1>` als Beispiel (Abbildung 2.2 und 2.3). Obwohl der Benutzer den Text „Das ist eine Überschrift“ in

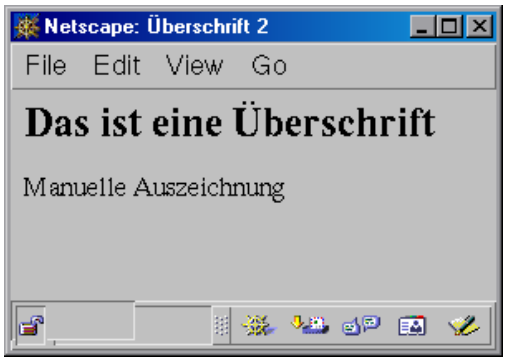



```
<HTML>
  <HEAD>
    <TITLE>Überschrift 1</TITLE>
  </HEAD>
  <BODY>

  <H1>Das ist eine Überschrift</H1>

  <P>Logische Auszeichnung
  </BODY>
</HTML>
```

Abbildung 2.2: HTML – Logische Formatierung




```
<HTML>
  <HEAD>
    <TITLE>Überschrift 2</TITLE>
  </HEAD>
  <BODY>
  <FONT size=+3>
    <B>Das ist eine Überschrift</B>
  </FONT>
  <P>Manuelle Auszeichnung
  </BODY>
</HTML>
```

Abbildung 2.3: HTML – Manuelle Formatierung

beiden Dateien gleich sieht, ist der Inhalt der Dateien unterschiedlich. Wie soll nun ein Agent solche Änderungen des „Inhaltes“ bewerten? Wie erkennt ein Agent, was der Benutzer mit „Inhalt“ meint?

Im Falle von HTML-Dokumenten gibt es außer dem eigentlichen Text auf der Seite noch einige andere „Inhaltsträger“, die ein Agent zur Analyse benutzen kann. In HTML-Seiten befinden sich meist Links auf andere Dokumente im WWW, viele Seiten im Web sind also durch Hyperlinks miteinander „verbunden“. Ein Agent kann den Aufbau der URLs ausnutzen, um bessere Vergleiche zwischen Seiten durchführen zu können. So ist die Ähnlichkeit von http://www.foo.com/bar/bar.html zu http://www.foo.com/bar/info.html sicherlich größer anzusehen als zu http://www.bar.info/bar/info.html.
Auch für die Hyperlinks kann man die unterschiedlichen Gebrauchsarten anwenden. So kann man den Text eines Hyperlinks dessen attributiver Verwendung zuordnen, dem referentiellen Gebrauch entspricht die Link-URL.

Auch Ähnlichkeiten in der inhaltlichen Struktur könnten von Nutzen sein. So ändert sich der „Inhalt“ von Newstickern zwar oft, der strukturelle Aufbau der Seite bleibt aber mehr oder weniger konstant. Ein Agent könnte also versuchen, die Struktur des Dokumentes zur Analyse auszunutzen. Leider ist die Analyse und der Vergleich von Strukturen sehr schwierig und aufwendig.

Aufgrund der unterschiedlichen Gebrauchsarten (Kapitel 2.1) stellt sich zusätzlich die im Anfang des Kapitels genannte Frage: „Welche Gebrauchsart wird verwendet?“. Wird das Dokument attributiv oder referentiell genutzt?
Im Falle des attributiven Gebrauchs sollten Änderungen stärker berücksichtigt werden als bei der referentiellen Verwendung[4].

2.3 Fehlerarten

Im WWW gibt es keine Garantie für den Benutzer auf eine bestimmte Ressource zugreifen zu können. Es gibt viele verschiedene Fehler unterschiedlicher Natur, die auftreten können, bevor der Inhalt von http://www.foo.com/bar/foo.html bei dem Benutzer angekommen ist. Einige Fehler sind dabei nur zeitweise (z.B. Serverausfall), andere sind permanent (z.B. Inhalt verschoben). Außerdem gibt es klare Unterschiede zwischen Seitenfehlern und inhaltlichen Fehlern. Bei Fehlern der ersten Kategorie hat ein Agent erst gar keinen Zugriff auf den „neuen“ Inhalt. Zudem sind Seitenfehler viel einfacher zu erkennen und handhaben, da die Bedeutungen zum großen Teil im Protokoll vorgegeben sind. Da kein Seiteninhalt vorhanden ist, tritt das Problem der Gebrauchsart auch praktisch nicht auf.

Die gängigen Linkchecker (siehe Kapitel 3.1) beschränken sich darauf Seitenfehler zu erkennen. In keinem Falle handhaben diese Programme inhaltliche „Fehler“, der aktuelle

[4] Bei der die Dokument-URL das Ausschlaggebende ist

und vergangene Inhalt der überprüften Seiten wird nicht bewertet, sondern gegebenenfalls lediglich dazu benutzt eine rekursive Überprüfung einer Seite zu ermöglichen.

2.3.1 Seitenfehler

Diese Fehlerkategorie bezieht sich auf Fehler, die auftreten können bevor der Inhalt der `URL` an den Benutzer übermittelt wurde. Tritt einer dieser Fehler auf, so ist eine inhaltliche Bewertung bzw. Überprüfung nicht möglich, da zu diesen Zeitpunkt noch kein neuer „Inhalt“ vorhanden ist. Die Seitenfehler lassen sich in Netzwerkfehler und Zugriffsfehler aufteilen. Bei den Netzwerkfehlern ist erst gar kein Zugriff auf den Server möglich, während die Verbindung bei den Zugriffsfehlern aufgebaut werden konnte.

2.3.1.1 Netzwerkfehler

In dieser Kategorie befinden sich alle Fehler, die aufgrund von Fehlern im Netzwerk auftreten können. Die folgenden zwei Fehler sind hier repräsentativ vorgestellt, weitere Fehler können auftreten, die Unterscheidung ist aber im Rahmen dieser Arbeit nicht relevant:

- Bei einem Zugriff auf den Server `www.foo.com` muß dieser Name erst einmal per DNS[5]-Dienst aufgelöst werden, d.h. es wird die IP-Adresse des Servers ermittelt. Durch diese Anfrage wird die Adresse des Objektes mit dem Namen `www.foo.com` ermittelt[6]. Schon diese Anfrage kann nicht erfolgreich sein, mit der Konsequenz das der Zugriff auf `www.foo.com` nicht möglich ist. Dieser Fall ist vergleichbar mit einer Anfrage an die Telefonauskunft: Frage ich bei der Auskunft nach der Telefonnummer von „Herrn Schmidt“, erhalte aber keine positive Antwort, so kann ich auch nicht mit „Herrn Schmidt“ telefonieren.
- Nachdem der Name des Servers erfolgreich ermittelt wurde, kann jetzt versucht werden den Kontakt zu `www.foo.com` aufzunehmen. Auch hier gibt es mehrere mögliche Ursachen, an denen der Verbindungsaufbau scheitern kann:
  1. Der Weg zu `www.foo.com` kann unterbrochen sein, es kann zum Beispiel sein das ein Router auf dem Weg zu `www.foo.com` ausgefallen ist.
  2. Der Server `www.foo.com` selbst ist nicht verfügbar, weil er zum Beispiel aufgrund eines Defektes in der Stromversorgung ausgefallen ist oder gerade wegen Wartungsarbeiten „heruntergefahren“ wurde. Außerdem ist es auch möglich,

[5] Domain Name Service
[6] Zur Klärung von „Name“ und „Adresse“ siehe Kapitel 2.2.1

das ein überlasteter Server keine Antwort liefert und die Verbindung wegen einer Zeitüberschreitung nicht zustande kommt.

2.3.1.2 Zugriffsfehler

Ist die Verbindung endlich zustande gekommen, so gibt es immer noch Gründe, die verhindern das der gewünschte Inhalt an den Benutzer übermittelt wird. Alle hier behandelten Fehler beziehen sich auf das HTTP-Protokoll ([Fielding et al., 1999]), da sich der in dieser Arbeit entwickelte Agent – momentan – auf die Verarbeitung von HTML-Seiten beschränkt. In anderen Protokollen ist eine andere Fehlerzuordnung möglich.

Im HTTP-Protokoll sendet der Benutzer (Client) eine Anfrage (Request) an den Server und bekommt daraufhin eine Antwort (Response), die einen Statuscode enthält. Anhand dieses Statuscodes kann der Client den Fehler ermitteln und das weitere Verhalten anpassen. Die folgenden Fehler verhindern unter anderem eine erfolgreiche Übermittlung des Inhaltes der angeforderten `URL`:

1. Serverfehler
   In diese Kategorie fallen alle Protokollfehler, die unabhängig von der angeforderten Seite auf dem Server auftreten können und deren Ursache im Server begründet liegt. Im HTTP-Protokoll sind diese als „Server Error“ bezeichnet und haben einen Statuscode von 5XX. Als Beispiele sind hier zu nennen:

   - 500 Internal Server Error
     Der Server hat einen unerwarteten Fehler erkannt, wegen dem er die Anforderung nicht abarbeiten kann. Dies kann z.B. passieren, wenn ein CGI-Skript eine nicht standardkonforme Antwort liefert, oder eine unerwartete Exception in einem JAVA-Servlet auftritt.

   - 503 Service Unavailable
     Der Server kann die Anforderung zur Zeit nicht bearbeiten, weil er überlastet ist, oder gewartet wird. Obwohl die Fehlerursachen die gleichen sein können, wie im vorhergehenden Abschnitt über Netzwerkfehler, ist der Server hier doch zumindest in der Lage, dem Client den aktuellen Status mitzuteilen. Wenn schon ein Fehler dieser Art auf dem Server auftritt, ist dies die bessere Vorgehensweise, den Client über den Fehler zu informieren. Der Client wartet im anderen Falle erst auf eine Zeitüberschreitung, die im Vergleich zur Fehlerantwort sehr viel Zeit beansprucht. Allerdings ist dieses Verhalten im Falle einer Überlastung nicht vorgeschrieben, so daß sich ein Client nicht darauf verlassen kann, bei einer Serverüberlastung einen 503-Fehler zu erhalten.

- 501 Not Implemented
  Der Server unterstützt die gewünschte Funktionalität nicht, um die Anforderung zu erfüllen. Dies tritt z.B. auf, wenn der Server die vom Client benutzte Anforderungsmethode nicht unterstützt.

2. Seitenbezogene Fehler
   Diese Fehler drücken ein Problem mit der angeforderten Seite des Servers aus. Der Server an sich hat keine Fehlfunktion, vielmehr liegt der Fehler aus der Sicht des Servers bei dem Client. Oft auftretende Fälle sind:

   - 401 Unauthorized
     Der Client ist nicht berechtigt, die angeforderte Seite ohne Anmeldung zu erhalten. Um auf die Seite zuzugreifen ist also zum Beispiel eine Anmeldung mit Benutzername und Passwort nötig. Hat der Client schon Anmeldeinformationen in der Anforderung mitgesendet, so hat er Server entweder die Anmeldemethode abgelehnt, oder die Anmeldeinformationen sind nicht korrekt.

   - 403 Forbidden
     Der Server hat die Anforderung zwar akzeptiert, verweigert dem Client aber die Antwort auf die Anfrage. Auch nach einer Anmeldung an den Server wird sich diese Antwort nicht ändern, der Client braucht also die Anmeldung im Gegensatz zum 401-Fehler erst gar nicht versuchen. Oft wird diese Meldung vom Server benutzt, wenn der Client keine genauen Informationen über die Ursache des Fehlers bekommen soll, oder wenn kein anderer Fehler „passt“.

   - 404 Not Found
     Die angeforderte Seite wurde auf dem Server nicht gefunden. Im Gegensatz zu der 301/302-Fehlermeldung ist kein Hinweis auf den neuen Ort der Seite verfügbar und es ist auch nicht bekannt, ob dieser Zustand nur temporär oder permanent ist.

   Weiterhin gibt es noch seitenbezogene Fehler, die auf eine Verschiebung des Inhaltes – sei es temporärer oder permanenter Art – hinweisen. Bei diesen Fehlern kann die zusätzliche Information in der Antwort des Servers von Client benutzt werden, um den neuen „Ort“ des Inhaltes zu ermitteln und so doch noch an den gewünschten Inhalt zu kommen. Zwei wichtige Arten der Weiterleitung per HTTP-Protokoll sind:

   a) 301 Moved Permanently
      Die Seite wurde permanent verschoben und ist unter der in der Antwort gelieferten `URI` zu finden. Der Client sollte alle zukünftigen Anfragen zu dieser Seite mit der neuen `URI` stellen.

b) 302 Moved Temporarily
Die Seite ist zur zeitweise unter der in der Antwort gelieferten `URI` zu finden. Da diese Umleitung unter Umständen nur kurzzeitig gültig ist, sollte der Client für zukünftige Anfragen weiterhin die alte `URI` benutzen.

2.3.2 Inhaltliche Fehler

Nachdem der Benutzer endlich den Inhalt der Seite erhalten hat, also keine Fehler aus Kapitel 2.3.1 aufgetreten sind, so stellt sich anschließend die Frage, ob der „neue“ Inhalt der Seite immer noch dem „alten“ Inhalt der Seite entspricht. In Kapitel 2.2.2 wurde die Problematik von „entsprechen“ schon aufgegriffen, für den Agenten lassen sich drei Fehlerarten unterscheiden. Der für den Inhalt vierte Fall, nämlich der „gleiche Inhalt“ ist in diesem Sinne ja kein Fehler und wird hier deshalb auch nicht explizit aufgeführt.

2.3.2.1 ähnlicher Inhalt

Der neue Inhalt hat sich gegenüber dem alten Inhalt geändert, ist aber noch zum großen Teil gleich geblieben. Hier stellen sich sofort wieder die Fragen: „Was ist eine Änderung?“ und „Wann ist eine Änderung *klein*?“
Sollen zum Beispiel geänderte Links oder umsortierte Abschnitte in die Ähnlichkeitsbewertung einfließen? Ist der gesamte Inhalt des Dokumentes oder etwa nur die Struktur wichtig? Die Antwort auf diese Fragen kann man nur durch den Benutzer und sein Verhalten annähern. Ein Agent kann nur versuchen, sein Verhalten durch Anwendung maschinellen Lernens an das Benutzerverhalten anzunähern, eine allgemeingültige Aussage wann eine Änderung groß ist wird es nicht geben.

2.3.2.2 unterschiedlicher Inhalt

Der neue Inhalt ist zum großen Teil geändert worden, oder hat gar nichts mehr mit dem alten Inhalt zu tun. Hier lassen sich die Ergebnisse aus dem vorhergehenden Abschnitt anwenden. Die Frage „Wann ist eine Änderung *groß*?“ läßt sich schließlich auch folgendermaßen formulieren: „Wann ist eine Änderung *nicht* mehr *klein*?“ Das Problem kann also auf die Beantwortung des vorhergehenden Problems zurückgeführt werden.

2.3.2.3 Weiterleitung

Der alte Inhalt ist zwar nicht mehr hier verfügbar, es sind aber Informationen im neuen Inhalt verfügbar, wo der alte Inhalt jetzt zu finden ist. In dieser Kategorie gibt es verschiedene Fälle, die dem Benutzer alle nahezu gleich schwer vorkommen, aber für den Agenten große Unterschiede in der Schwierigkeit darstellen. Teilweise bemerkt der Benutzer die Weiterleitung gar nicht erst weil diese schon stattfindet, bevor überhaupt ein Inhalt zu sehen war.

Die grundsätzliche Unterscheidung der Fälle kann durch die Art und Weise getroffen werden, in der die Informationen über den neuen Ort im aktuellen Inhalt vorhanden sind. Die Weiterleitungen des HTTP-Protokolls sind hier nicht zu betrachten, da sie ja nicht im Inhalt des Dokuments vorhanden sind, sondern schon vorher auftreten.
Bei HTML-Seiten gibt es genormte Möglichkeiten, eine Weiterleitung kenntlich zu machen und es so ermöglichen, diese maschinell auszuwerten. Gemeint ist hier die Weiterleitung im Dokument per HTML-Meta-Tag (`<meta http-equiv="refresh" ...>`), die in [Berners-Lee, D.Conolly, 1995] spezifiziert ist.

Ist die Information über den neuen Ort jedoch frei im Text vorhanden, so wird die Auswertung für einen Agenten sehr schwierig, da keine festen Muster zur Erkennung der Information vorhanden sind. Der Agent müßte also die Informationen erkennen und von anderen Elementen wie zum Beispiel Links unterscheiden. Dazu wäre aber Wissen über die Semantik des Inhaltes nötig, die für den Menschen relativ einfach erkennbar ist, aber sehr schwer fassbar für den Agenten. Eine Möglichkeit bestünde darin, daß der Agent versucht aus Beispielen bestimmte Muster zu lernen, anhand derer er die Weiterleitungs-Informationen erkennen kann, dies setzt aber schon voraus, das überhaupt feste Muster existieren.

Kapitel 3

Lösungen

Nachdem die Probleme, die auf den Benutzer bzw. Agenten zukommen können im vorhergehenden Kapitel geschildert wurden, sollen in diesem Kapitel Lösungsansätze, Hilfsmittel und Techniken behandelt werden, die geeignet erscheinen, diesen Problemen zu begegnen oder zur Realisierung des Agenten genutzt werden können. Hier noch mal ein kurzer Überblick über die Probleme:

- Benutzerverhalten
  Wie handhabt der Benutzer die Seiten? Sind *gleiche* Seiten erforderlich, reichen *ähnliche* Seiten, oder sind Seiten *der gleichen Art* akzeptabel?
- Erreichbarkeit
  Ist der Inhalt der `URL` noch verfügbar? Sollte dies nicht mehr der Fall sein, ist der Fehler dann temporär oder permanent?
- Inhalt
  Ist der Inhalt noch der gleiche? Mit welchen Mitteln kann der Agent den Unterschied zwischen Dokumenten feststellen?

In den folgenden Abschitten werden zunächst mögliche Lösungsansätze diskutiert, anschließend werden Techniken vorgestellt, die zur Entwicklung des Agenten benutzt werden können.

3.1 LinkChecker/Web Spider

LinkChecker prüfen einzelne Seiten oder ganze Websites auf die Konsistenz der vorhandenen Links, ausgehend von einer Startseite werden alle Links verfolgt und gegebenenfalls rekursiv geprüft. Konsistenz bedeutet hier lediglich das Vorhandensein des Inhaltes, der Link bzw. die `URL` wird nicht anhand ihres Inhaltes bewertet. Der Sinn und Zweck von LinkCheckern und Webspidern ist nun mal nur die Überprüfung, ob der Inhalt erreichbar

ist. Wenn überhaupt Änderungen erkannt werden, sind diese Änderungen zum Beispiel anhand des Datums der Seite erkannt worden.

LinkChecker haben als Hauptaufgabe, die Überprüfung auf defekte Links vorzunehmen. Im Vergleich dazu ist das Ziel bei Webspidern Seiteninhalte herunterzuladen, um diese dann einer Datenbank oder ähnlichem hinzuzufügen. Google und Altavista benutzen zum Beispiel selbst entworfene Webspider um Seiten aus dem Internet zu indexieren. Es gibt aber auch Webspider für den Heimanwender, wie z.B. Teleport Pro.

Zusammenfassend ist also zu sagen, daß LinkChecker nur dazu benutzt werden können, Seitenfehler (siehe Kapitel 2.3.1) zu erkennen. Die Erkennung von inhaltlichen Fehlern ist mit Programmen dieser Art nicht möglich.

3.2 Softwareagenten

Im folgenden wird ein Überblick über verschiedene Agenten gegeben. Dabei soll auf Ähnlichkeiten und Unterschiede im Bezug auf den in dieser Arbeit entwickelten Agenten eingegeangen werden.

3.2.1 WebWatcher

WebWatcher[Armstrong et al., 1995],[Joachims et al., 1997] ist ein Tour Guide Agent, der Benutzer beim Browsing auf dem `WWW` unterstützt. Anhand der Benutzerentscheidungen (gewählte Links) lernt WebWatcher das nötige Wissen um Touren zu geben automatisch. Im Gegensatz zum `URLChecker` ist er ein Online-Agent, d.h. der Benutzer ist während des Einsatzes des Agenten aktiv. WebWatcher benutzt die Linktexte der vom Benutzer gewählten Verweise, um sein Wissen zu erweitern. Durch den Einsatz maschinellen Lernens benutzt er dieses Wissen, um spätere Entscheidungen zu treffen. Diese Entscheidungen präsentieren sich dem Benutzer als Hinweise, für WebWatcher als interessant eingestufte Links werden mit einer vorangestellten Grafik hervorgehoben. Der Benutzer kann so durch seine Tour gelenkt werden. Auch `URLChecker` kann aus dem gesammelten Wissen lernen, allerdings geschieht dies nicht Online, sondern Offline. Desweiternen wird – im Gegensatz zu WebWatcher – der gesamte Inhalt der Seite von `URLChecker` als „Wissen“ verwendet.

Der große Unterschied besteht in den unterschiedlichen Einsatzzielen der Agenten: WebWatcher soll den Benutzer während einer Tour begleiten und für das Thema relevante Links hervorheben, im Gegensatz dazu soll `URLChecker` vom Benutzer gegebene Seiten auch ohne Benutzerinteraktion auf Änderungen überwachen. Diese unterschiedlichen Zielsetzungen erfordern natürlich auch unterschiedliche Vorgehensweisen.

3.2.2 Bibliothek-Agent von C. Bordihn

Der von C. Bordihn in Rahmen einer Diplomarbeit entwickelte Agent [Bordihn, 2000] soll dem Benutzer, in diesem Falle in der Rolle eines Bibliothekars, bei der Erstellung einer Dokumentdatenbank behilflich sein. Unter anderem sollen gefundene Dokumente in eine bestehende Klassenhierarchie eingeordnet werden. Zur Vorabklassifikation gefundener Dokumente wird eine hierarchische Klassifikation verwendet, die sich aufgrund der gegebenen Klassenstruktur anbietet. Dem Bibliothekar werden dann die Dokumente gemäß der Vorabklassifikation angeboten, der die Dokumente dann letztendlich in die Hierarchie einsortiert oder verwirft. Normale Benutzer können dann in der Hierarchie navigieren und sich die einsortierten Dokumente ansehen.

Der Agent benutzt die Textklassifikation zur Einordnung der Texte in die gegebene Hierarchie, im Gegensatz zu `URLChecker` ist diese Hierarchie aber schon vorgegeben, d.h. der Agent kann Klassifikatoren für feste Themengebiete trainieren. Dies ist für `URLChecker` nicht möglich, da sich die überwachten Seiten nicht notwendigerweise in dem selben Themengebiet befinden müssen. Bordihns Agent vergleicht Dokumente also anhand der Themenähnlichkeit. Deshalb ist eine „direkte“ Benutzung der Dokumente zum Lernen möglich.

3.3 Lernverfahren

Wie schon in Kapitel 2 klar wurde, wird die Benutzung von Verfahren maschinellen Lernens durch den Agenten notwendig sein, um die Bewertungen dem Benutzerverhalten anpassen zu können. Zunächst muss die Repräsentation der Daten geklärt werden. Welche Informationen können erfolgreich genutzt werden um zu Lernen?

Eine mögliche Formalisierung des Lernens ist die Suche einer Funktion. Die Aufgabenstellung ist, zu einer gegebenen Menge von Beispielen $(x_i, y_i) \in X \times Y$ eine Funktion $f : X \rightsquigarrow Y$ zu finden, die ein vorgegebenes Qualitätskriterium erfüllt. Dabei ist X die Menge der Beipiele und Y die Menge der möglichen Klassen zu denen das Beispiel gehören kann. Im Falle der Klassifikationsaufgabe gibt es genau zwei Klassen.
Andere Arbeiten benutzen zur Klassifikation der Beipiele die Textklassifikation in unterschiedlichen Weisen. Eine bewährte Methode beruht auf dem sogenannten Vektorraummodell [Salton, 1991], in dem die Dokumente als Wortvektoren repräsentiert sind. Anhand dieser Wortvektoren wird dann die Ähnlichkeit zweier Dokumente bestimmt. Eine Möglichkeit ist hier die Verwendung des Cosinus-Maßes, welches den Winkel zwischen den beiden Vektoren ermittelt. Dabei wird davon ausgegangen, das die Vektoren ähnlicher Dokumente einen geringen Winkel zueinander besitzen.

Im Falle des `URLChecker`s stellt sich die Frage, ob die Wortvektoren der Dokumente ebenfalls erfolgreich genutzt werden können. Das Problem liegt in den großen Unterschieden, die zwischen den überwachten Seiten bestehen können. Die sinnvolle Verwendung des Cosinusmaßes basiert auf der Ähnlichkeit der Vektoren bei ähnlichen Dokumenten. Dies kann aber z.B. im Falle einer überwachten Seite mit dem aktuellen Kinoprogramm nicht so sein. Welche anderen Möglichkeiten gibt es also noch?

Da die Verwendung eines einzigen Ähnlichkeitsmaßes im allgemeinen nicht ausreichen wird, sollten mehrere Ähnlichkeitsmaße kombiniert werden, um die unterschiedlichen Verhaltensweisen des Benutzers anzunähern. Aber wie sollen die Maße sinnvoll kombiniert werden? Es bestehen zwei Möglichkeiten:

1. Feste Kombination der Maße, Lernen über der Kombination
   Die Maße werden nach einem fest vorgegebenen Schema kombiniert. Das Ergebnis der Kombination wird dann zur Klassifikation genutzt. Problem ist hier zum einen die geringe Aussagekraft des kombinierten Maßes, zum anderen stellt sich die Frage nach der Herkunft der Berechnungsvorschrift zu Kombination. Es ist zu bezweifeln, das *eine einzige* Berechnungsvorschrift existiert, die alle Verhaltensweisen erfolgreich kombinieren kann.

2. Lernen der Kombination der Maße
   Die Maße werden zunächst unabhängig betrachtet. Für einen Vergleich zweier Dokumente wird ein Vektor aus den Ergebnissen der Distanzmaße gebildet. Dieser Vektor wird dann als Ausgangsbasis für die Entscheidung in Bezug auf die Ähnlichkeit der Dokumente benutzt.

Hier ist die Verwendung der zweiten Möglichkeit vorzuziehen. Durch das Lernverfahren wird dann sozusagen die Gewichtung der Ähnlichkeitsmaße anhand der Beispiele gelernt. Ein für diese Lernaufgabe gut einzusetzendes Verfahren ist die Support Vector Machine (kurz SVM), die gleich genauer vorgestellt wird. Mit Hilfe der SVM wird zuerst ein Modell gelernt, anschließend können Vergleichsvektoren anhand des Modells klassifiziert werden.

Durch die Benutzung der Ähnlichkeitsvektoren zum Lernen wird im Prinzip über das Ähnlichkeitsmaß gelernt und nicht über die Dokumente selbst. So ergibt sich eine größere Unabhängigkeit der Bewertung von den konkreten Dokumenten, welche sich im Falle des `URLChecker`s ja sehr stark unterscheiden können. Durch das Lernen des Ähnlichkeitsmaßes kann sich der Agent im Gegensatz zur festen Kombination relativ gut an das Benutzerverhalten anpassen.

3.3.1 Support Vector Machine

Dieser Abschnitt behandelt die Support Vector Machine (SVM), deren Theorie im Bereich des statistischen Lernens begründet liegt. Die grundlegenden Theorien der SVM sind schon seit geraumer Zeit bekannt, allerdings erfreut sich die SVM in letzter Zeit einer hohen Popularität.

SVMs basieren auf dem Prinzip der strukturellen Risikominimierung und sind gut dazu geeignet hochdimensionale Daten zu verarbeiten. Neben der Lösung der Klassifikationsaufgabe kann man SVMs auch zur Regression benutzen, in dieser Arbeit wird die SVM aber ausschließlich zur Klassifikation genutzt. Eine ausführliche Einführung in die Theorien der SVM ist in [Burges, 1998] zu finden. Aufbauend auf diesem Tutorial werden im folgenden die Grundlagen der Theorie dargestellt.

3.3.2 Das Lernproblem und Risiko

Für die SVM gelten noch weitere Einschränkungen bezüglich der Definition des Lernproblems, da es ansonsten nicht möglich wäre dieses zu lösen.
Vorausgesetzt sei eine Menge von Beispielen X. Jedes Beispiel besteht aus einem Vektor $\vec{x}_i \in \Re^n$ und der Klasse y_i, die als korrekte Klasse des Beispiels angenommen wird. Desweiteren wird angenommen, das eine Wahrscheinlichkeitsverteilung $P(\vec{x}, y)$ existiert, die für die Erzeugung der Daten zugrundegelegt werden kann. Die Beispiele sollen dabei als unabhängig voneinander und identisch verteilt betrachtet werden. $P(\vec{x}, y)$ selbst muß dabei nicht bekannt sein.
Angenommen, die Aufgabe der Maschine ist, die Abbildung $\vec{x}_i \rightsquigarrow y_i$ zu finden, dann ist die Maschine durch die Menge der möglichen Funktionen $f(x, \alpha)$ definiert. α sind hier die Parameter der Maschine. Die Maschine soll deterministisch sein, d.h. ein gegebenes x und α erzeugt immer die gleiche Ausgabe $f(x, \alpha)$.
Das *erwartete Risiko* ist dann:

$$R(\alpha) = \int \frac{1}{2}|y - f(x, \alpha)| dP(x, y) \tag{3.1}$$

$R(\alpha)$ ist der erwartete Testfehler der Maschine und wird auch einfach *Risiko* genannt.
Zusätzlich dazu gibt es noch das *empirische Risiko* $R_{emp}(\alpha)$, das als durchschnittlicher Fehler auf den Trainingsdaten definiert ist:

$$R_{emp}(\alpha) = \frac{1}{2|X|} \sum_{i=1}^{|X|} |y_i - f(x_i, \alpha)| \tag{3.2}$$

Die Größe $z = \frac{1}{2}|y_i - f(x_i, \alpha)|$ gibt die Abweichung zwischen dem vorhergesagten und realen Wert an und wird als Loss bezeichnet. Vapnik ([N.Vapnik, 1995]) definiert noch

eine obere Grenze für das erwartete Risiko. Vorausgesetzt η, z mit $0 \leq \eta \leq 1, 0 \leq z \leq 1$ ist die obere Grenze mit einer Wahrscheinlichkeit von $1 - \eta$ durch den folgenden Term beschränkt:

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\left(\frac{h(\log(2|X|/h) + 1) - \log(\eta/4)}{|X|}\right)} \tag{3.3}$$

h ist dabei eine positive, natürliche Zahl, die als Vapnik-Chervonenkis-Dimension (VC-Dimension) bezeichnet wird und die Ausdrucksstärke der Funktionenschar angibt. Der zweite Term auf der rechten Seite wird als „VC-Konfidenz“ bezeichnet.
Interessant ist, das die Berechnung der oberen Schranke unabhängig von der Wahrscheinlichkeit $P(x, y)$ ist. Obwohl die Berechnung von $R(\alpha)$ normalerweise nicht möglich ist, kann die rechte Seite der Gleichung leicht berechnet werden falls VC-Dimension h bekannt ist. Wenn man nun viele verschiedene Maschinen nimmt und dabei ein geeignet kleines aber festes η wählt, kann man die Maschine wählen, die die rechte Seite minimiert, um das maximale erwartete Risiko zu minimieren. Dieses Prinzip bezeichnet man als *strukturelle Rissikominimierung* (siehe Abschnitt 3.3.4).

3.3.3 Die Vapnik-Chervonenkis-Dimension

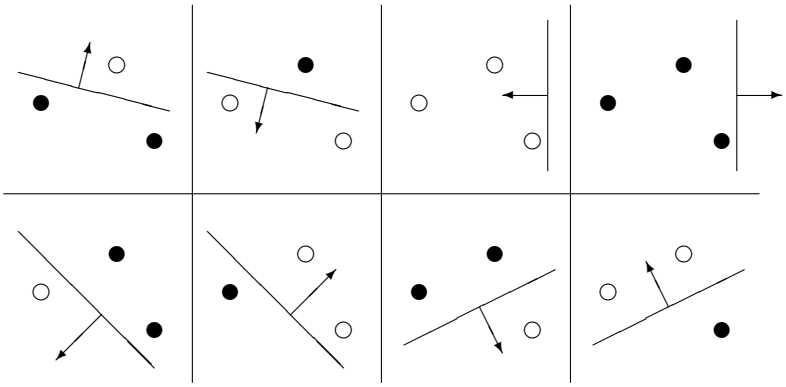


Abbildung 3.1: Lineare Trennungen von 3 Punkten im $\Re^2$

Die VC-Dimension ist eine Eigenschaft der Funktionsmenge $f(\alpha)$ und kann für verschiedene Funktionsklassen definiert weden. Für den hier aktuellen Fall der einfachen Klassifikation kann die Menge Y auf $-1, +1$ beschränkt werden. Gegeben eine Menge von l Punkten, können diese auf 2^l Weisen auf Y aufgeteilt werden. Wenn nun für jede dieser 2^l Mengen ein $f(\alpha)$ existiert, das diese Einteilung erfolgreich vorhersagt, wird die Menge der Punkte von der Menge der Funktionen *zerschmettert*. Die VC-Dimension ist nun definiert als die größte Anzahl von Punkten, die von $f(\alpha)$ zerschmettert werden kann. Das bedeutet, das bei einer VC-Dimension von h mindestens *eine* Menge von h

Punkten existiert die zerschmettert wird. Im allgemeinen wird aber nicht garantiert das *alle* Mengen mit h Punkten zerschmettert werden können.
Als Beipiel dient hier die Menge dreier Punkte im $\Re^2$ (Abbildung 3.1), die durch eine Geradenschar in alle möglichen Teilmengen aufgeteilt werden kann. Es ist kein Problem drei Punkte für eine gegebene Geradenschar zu finden, die beliebig durch diese getrennt werden kann, für vier Punkte ist dies allerdings nicht möglich. Die VC-Dimension für die Geradenschar im $\Re^2$ ist deshalb also 3. Im allgemeinen ist die VC-Dimension $h = n + 1$ für Hyperebenen im $\Re^n$.

3.3.4 Strukturelle Risikominimierung

Wie schon beschrieben besteht der Trainingsvorgang daraus, die Funktion aus $f(\alpha)$ zu finden, die das erwartete Risiko minimiert. Dabei ist zu beachten, das die VC-Konfidenz von der gewählten Funktionsklasse abhängt, das erwartete und empirische Risiko jedoch von der letztendlich gewählten Funktion. Das Prinzip der strukturellen Risikominimierung (SRM) basiert nun darauf, die Teilmenge der Funktionen zu finden, die die obere Grenze für das erwartete Risiko minimiert. Dazu wird die gesamte Klasse der Funktio-

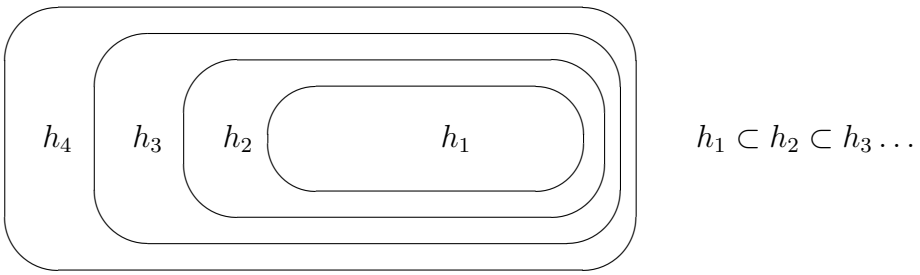


Abbildung 3.2: Struktur der Funktionsklassen

nen in eine Struktur geschachtelter Funktionsklassen aufgeteilt (Abbildung 3.2). Für jede Teilmenge muß nun die VC-Dimension h errechnet werden, oder zumindest eine obere Schranke von h. Durch das Trainieren mehrerer Maschinen (eine für jede Teilmenge), die jeweils das empirische Risiko minimieren, kann dann einfach die gesuchte Teilmenge ermittelt werden. Es wird die Maschine genommen, deren Summe aus empirischem Risiko und VC-Konfidenz minimal ist.

3.3.5 Lineare SVM

Wie gerade erläutert arbeiten SVMs gemäß dem Prinzip der strukturellen Risikominimierung. Die Frage die sich nun stellt ist, wie das Optimierungsproblem durch die SVM

gelöst wird. Wie findet die SVM die optimale Hyperebene? Es sei zunächst vereinfachend angenommen, die Trainingsmenge sei linear separierbar. Fehler und nicht-lineare Probleme werden später behandelt, die dann zu behandelnden Optimierungsprobleme sind aber dem linearen Fall sehr ähnlich. Die Klassifikation von unbekannten Beispielen erfolgt nach erfolgreichem Training durch Berechnung des Abstandes des Beispieles zur Hyperebene. Ist dieser positiv, so wird das Beipiel als zugehörig zur Klasse der positiven Beispiele betrachtet und umgekehrt.

Gegeben ist – wie gehabt – eine Menge von Trainingsbeispielen $\vec{x_i}, y_i, i = 1, \ldots, l, y_i \in -1, +1, \vec{x_i} \in \Re^d$ (l ist die Anzahl der Beispiele). Angenommen es gibt eine Hyperebene, die die positiven von den negativen Beipielen trennt, dann erfüllen die Punkte x auf der Hyperebene die Gleichung $\vec{w} \cdot \vec{x} + b = 0$, wobei w die Normale der Hyperebene ist. d_+ (d_-) bezeichnet den kürzesten Abstand von der Hyperebene zu einem positiven (negativen) Beispiel. Im linearen separierbaren Fall ermittelt die SVM genau die Hyperebene, deren Summe der Abstände maximal ist ($d_+ + d_-$). Dabei entspricht die Maximierung des Abstandes der Minimierung der Länge der Normalen, unter der Voraussetzung, das die folgenden Bedingungen erfüllt sind:

$$y_i(\vec{x}_i \cdot \vec{w} + b) - 1 \geq 0, \quad \forall i \tag{3.4}$$

Für den Fall des $\Re^2$ zeigt Abbildung 3.3 die erwartete Lösung. Die Beispiele die auf

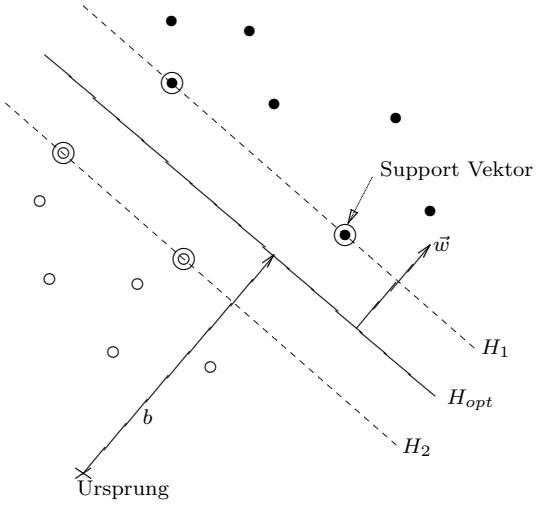


Abbildung 3.3: Linear separierende Hyperebene im $\Re^2$

dem Rand liegen, d.h. die den Gleichheitsfall in der obigen Ungleichung erfüllen, werden *Support Vektoren* genannt. Durch diese wird die Lösung der SVM bestimmt, alle anderen Beispiele werden für die Lösung nicht benötigt. Durch die Einführung von Lagrange-Multiplikatoren $\alpha_i, i = 1 \ldots l$ lassen sich die Bedingungen zur Maximierung des Randes in

die sogenannte Lagrange-Darstellung überführen. Der Vorteil dieser Darstellung besteht darin, das die Trainingsdaten nur in Form von Skalarprodukten auftauchen. Dies ist wichtig, um die Lösung später auf den nicht-linearen Fall erweitern zu können. Außerdem ist die Lagrange-Darstellung leichter zu berechnen.

$$L_P = \frac{1}{2}\|\vec{w}\|^2 - \sum_{i=1}^{l} \alpha_i y_i (\vec{x}_i \cdot \vec{w} + b) + \sum_{i=1}^{l} \alpha_i \tag{3.5}$$

Die Maximierung des Randes entspricht nun der Minimierung von L_P. Berücksichtigt man jetzt noch die dualen Bedingungen

$$\vec{w} = \sum_i \alpha_i y_i \vec{x}_i \quad \text{und} \quad \sum_i \alpha_i y_i = 0, \tag{3.6}$$

dann ergibt sich das folgende duale Optimierungsproblem L_D

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j \tag{3.7}$$

das dann maximiert werden muß. Somit hat sich ein quadratisches Programmierungsproblem ergeben, das mit Hilfe der *Karush-Kuhn-Tucker Bedingungen* (KKT) gelöst werden kann. Die KKT Bedingungen sind für die Lösung von $\vec{w}, b$ und α notwendig als auch hinreichend:

$$\frac{\partial L_P}{\partial w_\upsilon} = w_\upsilon - \sum_i \alpha_i y_i x_{i\upsilon} = 0, \quad \upsilon = 1, \ldots, d \tag{3.8}$$

$$\frac{\partial}{\partial b} L_P = -\sum_i \alpha_i y_i = 0 \tag{3.9}$$

$$y_i(\vec{x}_i \cdot \vec{w} + b) - 1 \geq 0, \quad i = 1, \ldots, d \tag{3.10}$$

$$\alpha_i \geq 0, \quad \forall i \tag{3.11}$$

$$\alpha_i(y_i(\vec{w} \cdot \vec{x}_i + b) - 1) = 0, \quad \forall i \tag{3.12}$$

Im Gegensatz zu b wird $\vec{w}$ explizit durch den Trainingsvorgang bestimmt. b kann aber implizit durch Gleichung (3.12) berechnet werden, indem ein i mit $\alpha_i \neq 0$ gewählt wird. b ergibt sich dann durch einfaches Ausrechnen der Gleichung.

Wie schon gesagt wurde für die Lösung die Separierbarkeit der Beispiele vorausgesetzt. Wie kann nun mit Fehlern in den Beispielen umgegangen werden? In der Realität wird man es im Regelfall mit inkonsistenten Daten zu tun haben, so daß eine strikte Trennung der Beipiele nicht mehr möglich ist. Diese Inkonsistenzen können z.B. durch Messfehler oder Rauschen verursacht worden sein.
Um nun das Problem auch im Falle von Fehlern zu lösen, wird ein zusätzliches Maß

$\xi_i, i = 1, \ldots, l$ mit $\xi_i \geq 0$ für den Fehler der Beispiele eingeführt. Durch Benutzung von ξ_i in Gleichung (3.4) kann diese im Falle von Fehlern etwas gelockert werden:

$$y_i(\vec{x}_i \cdot \vec{w} + b) - 1 \geq 1 - \xi_i, \quad \forall i \tag{3.13}$$

Damit die Berücksichtigung von Fehlern gesteuert werden kann ist es in diesem Falle sinnvoll, die Minimierung von $\|\vec{w}\|^2/2$ auf die Minimierung von $\|\vec{w}\|^2/2 + C(\sum_i \xi_i)^k$ zu erweitern, wobei C den Kosten eines Fehlers entspricht, d.h. ein größeres C bedeutet eine höhere Strafe für einen Fehler. Für $k = 1$ und $k = 2$ ergibt sich wieder ein quadratisches Programmierungsproblem, dessen Lagrange-Darstellung wie folgt ist:

$$L_P = \frac{1}{2}\|\vec{w}\|^2 + C\sum_i \xi_i - \sum_{i=1}^{l} \alpha_i \left(y_i(\vec{x}_i \cdot \vec{w} + b) - 1 + \xi_i\right) - \sum_{i=1}^{l} \mu_i \xi_i \tag{3.14}$$

Die Lagrange-Multiplikatoren μ_i wurden dabei eingeführt damit die ξ_i immer positiv sein müssen. Analog zum separierbaren Fall müssen auch wieder die KKT-Bedingungen gelten, die dann in Bezug auf ξ_i und μ_i erweitert bzw. angepasst werden.

3.3.6 Kernfunktionen

Eine zusätzliche Erweiterung der SVM besteht in der Benutzung von sogenannten Kernfunktionen. Bislang kann die SVM die Beispiele nur linear separieren, mit der Hilfe von Kernfunktionen wird diese Beschränkung aufgehoben. Die Trainingsdaten werden dazu mit Hilfe von $\Phi : \Re^d \mapsto \mathcal{H}$ in einen anderen euklidschen Raum transformiert, in dem dann wieder eine lineare Funktion gelernt werden kann. Die eigentlich gelernte Funktion ist dann aber nicht mehr linear.
Da die Trainingsdaten in allen Berechnungen der SVM nur in Form von Skalarprodukten auftauchen, muß die Transformation Φ nicht explizit bekannt sein, es reicht die Kenntnis einer Kernfunktion $K(\vec{x_i}, \vec{x_j}) = \Phi(\vec{x_i}) \cdot \Phi(\vec{x_j})$. In allen Gleichungen kann dann einfach jedes Skalarprodukt durch $K(\vec{x_i}, \vec{x_j})$ ersetzt werden. Für die Klassifikation ergibt sich die leicht geänderte Gleichung:

$$f(\vec{x}) = \text{sign}\left(\sum_{i=1}^{N_s} \alpha_i y_i K(\vec{s_i}, \vec{x}) + b\right) \tag{3.15}$$

Dabei sind die s_i die Support-Vektoren und N_s die Anzahl der Support-Vektoren der SVM.

Damit sich eine Funktion $K(\vec{x_i}, \vec{x_j})$ als Kernfunktion verwenden läßt, muß die *Mercer-Bedingung* erfüllt sein:
Es existiert ein Φ und

$$K(\vec{x_i}, \vec{x_j}) \tag{3.16}$$

genau dann, wenn für jedes $g(x)$ mit $\int g(x)^2 dx$ ist finit gilt:

$$\iint K(\vec{x_i}, \vec{x_j}) g(\vec{x_i}) g(\vec{x_j}) d\vec{x_i} d\vec{x_j} \geq 0 \tag{3.17}$$

Der Beweis kann dabei schwierig sein, da Gleichung (3.17) für *jedes* g mit finiter L_2-Norm gelten muß.

Die in dieser Arbeit genutzte SVM-Version (svm_{light}) unterstützt standardmäßig die folgenden Kernfunktionen:

$$\text{Linear} \quad : \quad K(\vec{x_i}, \vec{x_j}) = \vec{x_i} \cdot \vec{x_j} \tag{3.18}$$

$$\text{Polynomiell} \quad : \quad K(\vec{x_i}, \vec{x_j}) = ((\vec{x_i} \cdot \vec{x_j}) + 1)^d \tag{3.19}$$

$$\text{Radial} \quad : \quad K(\vec{x_i}, \vec{x_j}) = \exp^{-\gamma |\vec{x_i} - \vec{x_j}|^2}, \quad \gamma \in \Re_0^+ \tag{3.20}$$

$$\text{Sigmoid} \quad : \quad K(\vec{x_i}, \vec{x_j}) = \tanh(\alpha(\vec{x_i} \cdot \vec{x_j}) + b), \quad a, b \in \Re \tag{3.21}$$

Gleichung (3.20) entspricht dabei einem Klassifizierer mit einer Gausschen-Radialbasisfunktion, Gleichung (3.21) entspricht einem zweischichtigen neuronalen Netz.

3.4 Inhaltsvergleich

Der Vergleich von Seiteninhalten kann auf viele verschiedene Weisen erfolgen. Da anzunehmen ist, das der präzise Textvergleich schlechte Resultate liefern wird, soll hier ein Ähnlichkeitsmaß vorgestellt werden, das auf Levensthein ([Levensthein, 1965]) zurückgeht. Mit der *Editierdistanz* ist der ungenaue Vergleich von Inhalten möglich, d.h. Unterschiede zwischen den Inhalten resultieren in einem größeren Abstand. Die Editierdistanz ist dabei nicht auf Texte bzw. Zeichenketten beschränkt, der Algorithmus arbeitet vielmehr auf Sequenzen und benötigt lediglich eine Vergleichsfunktion, die (Un-)Gleichheit zweier Elemente bestimmen kann. So ist es z.B. möglich einen Buchstaben, ein Wort oder einen Satz als Einheit anzunehmen. Anwendungsgebiete finden sich zum Beispiel in Bereichen der Molekurarbiologie (Vergleich von DNA-Sequenzen), Spracherkennung oder auch der Rechtschreibkorrektur (Vorschlagen ähnlicher Wörter).

Die Editierdistanz $d(s_1, s_2)$ zweier Sequenzen s_1 und s_2 ist definiert als die *minimale* Anzahl von Änderungen, um s_1 in s_2 zu überführen. Eine Änderung kann dabei

1. das Austauschen eines Elementes,

2. das Einfügen eines Elementes, oder

3. das Entfernen eines Elementes

sein. $d(s_1, s_2)$ kann dann durch die folgenden Formeln beschrieben werden:

$$
\begin{aligned}
d([],[]) &= 0 && (3.22)\\
d(s,[]) &= \alpha|s| && (3.23)\\
d([],s) &= \beta|s| && (3.24)\\
d([e_1|s_1],[e_2|s_2]) &= \min(\; d(s_1,s_2) + \gamma\tau(e_1,e_2), && (3.25)\\
&\qquad\qquad\quad d([e_1|s_1],s_2) + \alpha,\\
&\qquad\qquad\quad d(s_1,[e_2|s_2]) + \beta \quad)
\end{aligned}
$$

Dabei sind e_1 und e_2 jeweils ein Element aus s_1 und s_2, $\tau(e_1, e_2)$ ist die Vergleichsfunktion für die Elemente und liefert 0, wenn die Elemente gleich sind und 1 sonst. Die Faktoren α, β und γ ermöglichen die Berücksichtigung unterschiedlicher Kosten für die verschiedenen möglichen Änderungen.
Gleichungen (3.22) bis (3.24) sind trivial, deshalb soll nur noch Gleichung (3.25) betrachtet werden. Beide Sequenzen sind in diesem Falle nicht leer, besitzen also ein erstes Element (e_1 bzw. e_2). Wenn die Elemente gemäß der Vergleichsfunktion τ gleich sind ($\tau(e_1, e_2) = 0$), dann kann der Vergleich der beiden Sequenzen ohne Strafe mit den Resten fortgesetzt werden, ansonsten gibt es drei Möglichkeiten:

1. Ersetzen des Elementes
   Es fallen die Kosten γ an. Der Vergleich kann mit s_1 und s_2 fortgesetzt werden.

2. Entfernen von e_1 aus s_1
   Es fallen die Kosten α an. Der Vergleich kann mit s_1 und $[e_2|s_2]$ fortgesetzt werden.

3. Einfügen von e_2 in s_1
   Es fallen die Kosten β an. Der Vergleich kann mit $[e_1|s_1]$ und s_2 fortgesetzt werden.

Aus diesen drei Möglichkeiten wird dann die „billigste“ Alternative gewählt. Die Berechnung der Editierdistanz kann offensichtlich als ein „Shortest-Path“-Problem aufgefasst werden.

Das Problem ist, wie man leicht erkennen kann, dreifach-rekursiv und die Laufzeit wäre somit exponientiell. Der Vergleich von Sequenzen mit mehr als ein paar Elementen wäre nicht möglich. Sieht man sich die Abhängigkeiten in Gleichung (3.25) aber genau an, so erkennt man das $d(s_1, s_2)$ ausschließlich von $d(s_1', s_2')$ abhängt, wobei s_1' (s_2') kürzer als s_1 (s_2) ist, oder aber beide. Dies ermöglicht die Benutzung von *dynamischer Programmierung* zur Lösung des Problems und führt zu einer Laufzeit von $O(|s_1| * |s_2|)$, was doch wesentlich besser als exponentielle Laufzeit ist.

Die Implementierung verwendet standardmäßig eine Matrix der Dimension $(|s_1|+1) \times (|s_2|+1)$ zur Speicherung der Distanzen. Wenn aber nur die Distanz benötigt wird und nicht die Sequenz der Änderungen, kann der Speicheraufwand auf $O(|s_1|+|s_2|)$ reduziert werden ([Hirschberg, 1975]).

Ein Beispiel
Der Algorithmus soll nun an einem Beispiel verdeutlicht werden. Nehmen wir einmal die beiden Worte „CHECKER“ und „TRACK“. Vereinfachend sei angenommen, das die Kosten für Ersetzen, Einfügen und Löschen gleich groß sind ($\alpha = \beta = \gamma = 1$).
Die Berechnung kann spaltenweise erfolgen, wobei sich der Wert der Zelle (i, j) durch das Minimum der drei Werte $(i-1, j-1) + \tau(i, j)$, $(i-1, j) + 1$, $(i, j-1) + 1$ ergibt. Gemäß Gleichungen (3.22) bis (3.24) ergeben sich zu Beginn des Algorithmus die Werte der ersten Zeile und Spalte.

		C	H	E	C	K	E	R
	0	1	2	3	4	5	6	7
T	1							
R	2							
A	3							
C	4							
K	5							

Startzustand

		C	H	E	C	K	E	R
	0	1	2	3	4	5	6	7
T	1	1	2					
R	2	2	2					
A	3	3	3					
C	4	3	4					
K	5	4	4					

Zustand nach zwei Durchläufen

		C	H	E	C	K	E	R
	0	1	2	3	4	5	6	7
T	1	1	2	3	4			
R	2	2	2	3	4			
A	3	3	3	3	4			
C	4	3	4	4	3			
K	5	4	4	5	4			

Zustand nach vier Durchläufen

		C	H	E	C	K	E	R
	0	1	2	3	4	5	6	7
T	1	1	2	3	4	5	6	7
R	2	2	2	3	4	5	6	6
A	3	3	3	3	4	5	6	7
C	4	3	4	4	3	4	5	6
K	5	4	4	5	4	3	4	*5*

Endzustand

Am Ende ergibt sich die Editierdistanz als Wert der Zelle in der unteren rechten Ecke der Tabelle (hier $(7, 5)$). Die Editierdistanz zwischen „CHECKER“ und „TRACK“ beträgt also 5.

3.5 BotIShelly

`BotIShelly` ([Banken et al., 2000]) ist ein im Rahmen einer Projektgruppe entwickelter Baukasten zur Agentenentwicklung. Durch den modularen Aufbau ist es relativ einfach

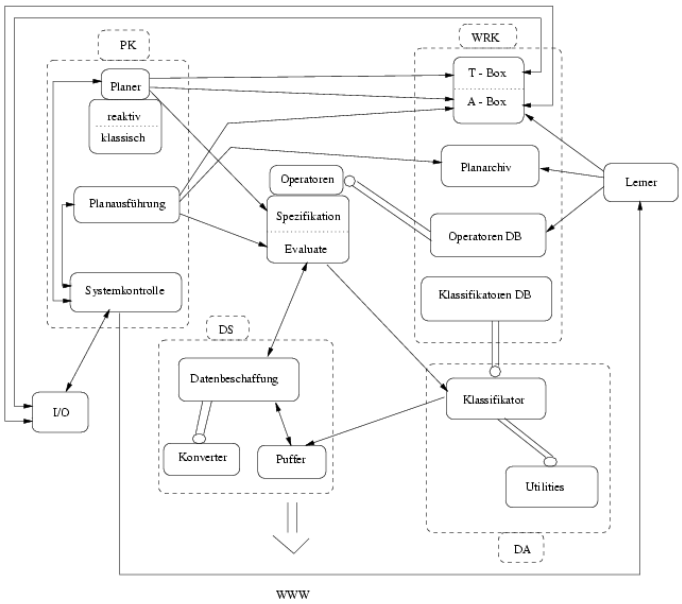


Abbildung 3.4: Architektur von `BotIShelly`

möglich neue Agenten zu entwickeln, die schon vorhandene Komponenten wiederverwenden. In [Malzahn, 2003] wurde `BotIShelly` z.B. zur Entwicklung eines Agenten benutzt (Pagetracker-Agent) der mit verschiedenen Strategien versucht, vorgegebene Seiten im `WWW` wiederzufinden.

Im Rahmen dieser Diplomarbeit wird `BotIShelly` auch genutzt. Die Klassen zur Erstellung und Benutzung von Dokumentvektoren sind ebenso im Gebrauch, wie auch die Komponente zum Zugriff auf das `WWW` (Datenbeschaffung). Der flexible Entwurf dieser Komponente ermöglicht einen, vom Dokumenttyp „unabhängigen“, Zugriff auf den Inhalt. So ist durch die Realisierung des PDFTextNetResult-Typs der einfache Zugriff auf den Text von PDF-Dokumenten realisiert worden. Nachteil ist aber die Beschränkung auf die unterstützten Typen. Da in dieser Arbeit weitestgehend mit HTML-Dokumenten, fällt dieser Nachteil nicht sehr ins Gewicht. Es überwiegt der Vorteil der vorhandenen Funktionen in Bezug auf HTML-Dokumente, so ist z.B. der einfache Zugriff auf die Links im Dokument möglich. Die textuelle Ausgabe des Inhaltes unterstützt Framesets automatisch, d.h. der Inhalt der Unterframes wird ebenfalls ausgegeben.

Die vorhandenen Klassen zur Umwandlung des Dokumentinhaltes in einen Dokumentvektor, sowie die Documentvector-Klasse selbst, vereinfachen den Umgang mit Dokumentinhalten ebenfalls erheblich. So ist zum Beispiel die Berechnung des Cosinusmaßes ohne weiteren Aufwand durch einen einfachen Methodenaufruf möglich. Mit Hilfe des

Vektorisierers kann der Inhalt eines Dokumentes durch frei konfigurierbare Trennzeichen in einen Dokumentvektor umgewandelt werden. Der Aufbau des dazu benötigten Wörterbuches (zur Zuordnung von Worte und Vektorpositionen) wird dabei gegebenenfalls automatisch erledigt.

Im Gegensatz zur eben genannten Diplomarbeit wird die Planer-Komponente für den `URLChecker` nicht benötigt und deshalb auch nicht verwendet. Der `URLChecker` kann aber als Operator in `BotIShelly` genutzt werden, ein solcher Operator wurde prototypisch erstellt. Die Einbindung des `URLChecker`s als Ähnlichkeitsmaß für den Seitenvergleich im Pagetracker-Agenten ist somit möglich. Ob die Ergebnisse dadurch verbessert werden können bleibt abzuwarten.
Auf den ersten Blick scheint die Verwendung der Klassifizierer-Architektur aus der Datenanalyse sinnvoll. Allerdings ist die Klassifizierer-Schnittstelle nicht für die Verarbeitung von Ähnlichkeitsvektoren ausgelegt. Desweiteren ist noch kein SVM-Klassifizierer realisiert worden, so daß die eigene Entwicklung eines Klassifizierers notwendig ist. Aus diesen Gründen wird auf die Verwendung dieser Komponente verzichtet.

Kapitel 4

Realisierung

Dieses Kapitel handelt von dem Agenten, der in dieser Arbeit entwickelt wird. Neben einer Übersicht über die Interaktion mit dem Benutzer sollen hier die Architektur und die eingesetzten Techniken verdeutlicht werden. Mit Hilfe der eingesetzten Techniken soll der `URLChecker` die in Kapitel 2 diskutierten Probleme behandeln.

Die momentane Benutzeroberfläche des `URLChecker`-Agenten basiert auf Servlets. Durch den Einsatz dieser Technik ist der Einsatz des Agenten auf Servern möglich, die Benutzung des Agenten kann jedoch ortsunabhängig per Browser erfolgen. Außerdem ist so die Steuerung des Agenten „von außen“ leicht möglich, indem zum Beispiel automatisierte Programme genutzt werden, die in regelmäßigen Zeitabständen Überprüfungen von `URL`s veranlassen.

Zunächst wird die Struktur von `URLChecker` beschrieben, um anschließend näher auf Schlüsselkomponenten des Agenten einzugehen. Anschließend wird der Arbeitsablauf von `URLChecker` an einem Beispiel verdeutlicht. Ein anderer Schwerpunkt dieses Kapitels ist die Vorstellung der von `URLChecker` verwendeten Distanzmaße, deren Performanz in Kapitel 5 ausgewertet wird. Der kombinierte Einsatz der unterschiedlichen Distanzmaße soll in Verbindung mit der SVM das Benutzerverhalten annähern und es `URLChecker` ermöglichen sich unterschiedlichen Verhaltensweisen anzupassen.

4.1 Architektur

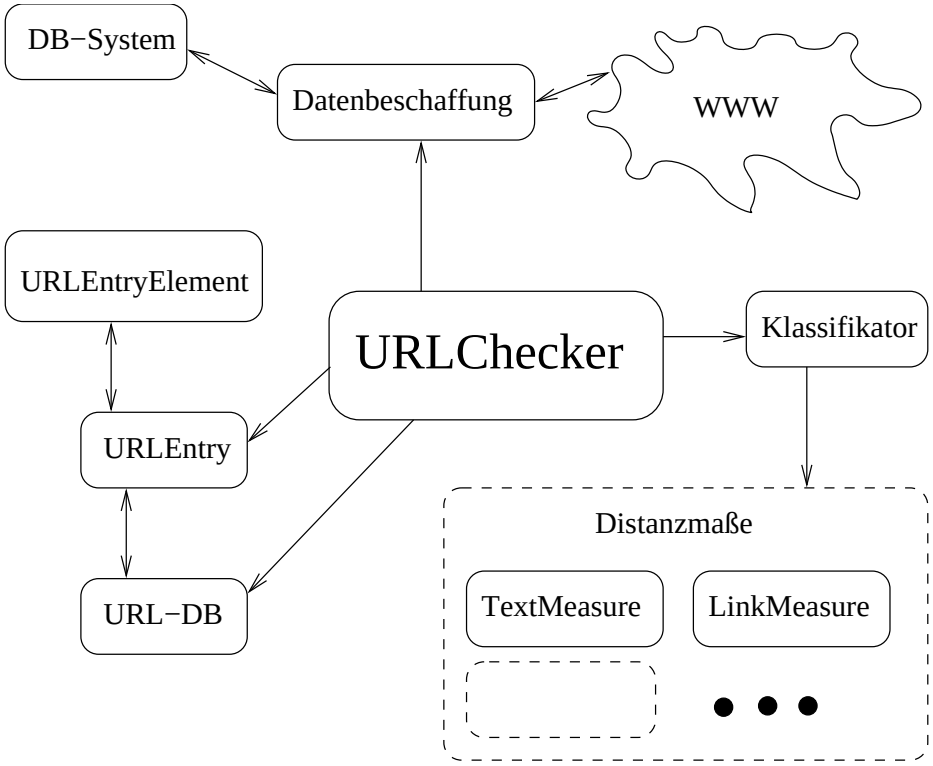


Abbildung 4.1: Architektur von `URLChecker`

Der Hauptteil des `URLChecker`s ist in der `URLChecker`-Klasse realisiert. Diese enthält die Grundfunktionalität des Agenten, sowie eine Schnittstelle, die in der Benutzeroberfläche sowie im Operator für `BotIShelly` genutzt wird. Durch die durchgehende Trennung von der Benutzerschnittstelle kann diese relativ einfach ausgetauscht werden.
Die einzelnen Benutzereinträge (`URL`s) sind als eigenständige Einträge (`URLEntry`) realisiert, in denen für jede Überprüfung der `URL` ein eigenes Element (`URLEntryElement`) erzeugt wird. Alle überwachten Einträge sind in einer Datenbank (`URLDB`) abgespeichert, die von der generischen Datenbank in `BotIShelly` abgeleitet ist.
Über die Datenbeschaffung greift `URLChecker` auf das World Wide Web zu um gegebene `URL`s zu überprüfen. Da die Datenmengen mitunter sehr groß werden können, ist die Datenbeschaffungskomponente unter Beihilfe von N. Malzahn um die datenbankgestützte Speicherung der Inhalte erweitert worden. Der Speicherverbrauch des Agenten kann dadurch mitunter erheblich reduziert werden, da die Inhalte nicht mehr alle im Hauptspeicher vorgehalten werden.

Ein weiterer wichtiger Bestandteil ist der Klassifikator in Verbindung mit der Gruppe der Distanzmaße. Durch die Bewertung der Distanzmaße durch den Klassifikator fällt `URLChecker` die Entscheidung über den Status des Inhaltes (siehe auch Kapitel 2.3.2). Dabei ist die Verwendung des Klassifikators so geschehen, das ein Austausch bzw. ein Wechsel des Klassifikators sehr einfach gemacht wird. Die verwendeten Maße sind in einer separaten Konfigurationsdatei festgelegt, und können zur Laufzeit festgelegt werden. Durch konsequenten Gebrauch einer Vererbungshierarchie bedeuted die Entwicklung zusätzlicher Maße nur einen geringen Aufwand.

4.2 Eintragsarten

- `http://www.foo.com/foo.html` ⟵ *URLEntry*
  - 12.10.2003
  - 27.9.2003 ⟵ *URLEntryElement*
  - 4.6.2003
  - 27.12.2002
- `http://www.bar.foo/`
  - 12.10.2003
  - 25.9.2003
  - 8.7.2003

Abbildung 4.2: Anordnung der Einträge im `URLChecker`

Für den `URLChecker`-Agenten gibt es zwei verschiedene Einträge:

1. `URLEntry`-Eintrag
   Der `URLEntry`-Eintrag repräsentiert eine überwachte `URL` und ist die Grundeinheit für `URLChecker`. Jeder Eintrag beinhalted die überwachte `URL`, den aktuellen Eintrag-Status, sowie alle für diesen Eintrag vorhandenen `URLEntryElemente`, die chronologisch angeordnet sind. Der Status bezeichnet hierbei, ob die Überprüfung gerade läuft, oder erfolgreich oder nicht erfolgreich war. Der differenzierte Status ist im jeweiligen Element vorhanden.
   Der `URLEntry`-Eintrag ist somit ein Behälter für alle Überprüfungen einer `URL`. Damit die Liste der Elemente nicht beliebig lang wird, kann die maximale Anzahl von Elementen beschränkt werden.

2. `URLEntryElement`-Eintrag
   Die Aufgabe des `URLEntryElement`-Eintrages ist die Kapselung einer Überprüfung der im übergeordneten Eintrag gegebenen `URL`. Jedes Element besitzt ein zugehöriges Datum (an dem die Überprüfung stattfand), die konkrete `URL` (kann z.B. bei Weiterleitung unterschiedlich sein), den Netzstatus des Inhaltes sowie den URLChecker-Status des Elementes (siehe Kapitel 4.3). Außerdem wird der Inhalt zum Vergleich mit anderen Versionen oder Seiten abgespeichert. Im Falle eines

Fehlers kann eine Meldung im Eintrag abgelegt werden.
Der `URLEntryElement`-Eintrag repräsentiert somit einen Schnappschuß einer `URL` zu einem bestimmten Zeitpunkt.

4.3 Stati

Für jedes Element gibt es einen URLChecker-Status S_e, durch den dessen Zustand ausgedrückt wird. Durch den Status sind zum Beispiel die einzelnen Fehlerarten (Netzwerkfehler, Seitenfehler, Inhaltsfehler) erkennbar, der Status des aktuellsten (zuletzt überprüften) Elementes beeinflußt dabei den Status S_u des gesamten Eintrages. Sollte ein Fehler aufgetreten sein, der nicht inhaltlicher Natur war, oder sollte sich der Inhalt geändert haben, so bekommt S_u als Status „Not OK“ zugewiesen, d.h. die Überprüfung war nicht erfolgreich. Ansonsten ist S_u „OK“ und die Überprüfung war erfolgreich. Am Status S_e des Elementes kann dann noch die genauere Bewertung der Seitenversion erkannt werden, `URLChecker` unterscheidet nämlich noch zwischen *ähnlich* und *gleich*. Tabelle 4.1 fasst die möglichen Werte von S_e zusammen.
Zur CONTENT_CLASS-Gruppe gehören alle Fehler inhaltlicher Natur, wie zum Bei-

CONTENT_CLASS	
CONTENT_DIFFERENT	CONTENT_NEW
CONTENT_EQUAL	CONTENT_MOVED
CONTENT_SIMILAR	CONTENT_UNSUPPORTED
NET_CLASS	
NET_NOT_FOUND	NET_UNREACHABLE
NET_TIMEOUT	
PROTOCOL_UNSUPPORTED	SYNTAX_URL

Tabelle 4.1: Mögliche Werte des URLCheckerstatus S_e

spiel unterschiedlicher oder fehlender Inhalt. CONTENT_NEW wird zugewiesen, wenn keine alte Version zum Vergleich vorhanden ist, also wenn die `URL` initial eingetragen wird. Netzwerkfehler sind in der NET_CLASS-Gruppe zu finden, dabei ist der Status NET_NOT_FOUND aber nicht mit dem HTTP-Fehler 404 zu verwechseln. Hier bedeutet dieser Status nämlich, das der Server – und nicht etwa die Seite – nicht gefunden wurde. Eine nicht gefundene Seite wird auf den Status CONTENT_MISSING abgebildet.
Am Ende der Tabelle sind dann noch zwei Stati aufgeführt, die in keine der anderen Gruppen passen. Zum einen ist es möglich, das die zu überwachende `URL` nicht wohlgeformt ist, der Eintrag bekommt dann den Status SYNTAX_URL. Die andere Möglichkeit ist zum Beispiel die Eintragung einer `mailto:-URL`. Diese wird von `URLChecker` nicht unterstützt, der Eintrag bekommt deshalb den Status PROTOCOL_UNSUPPORTED.

4.4 Arbeitsablauf

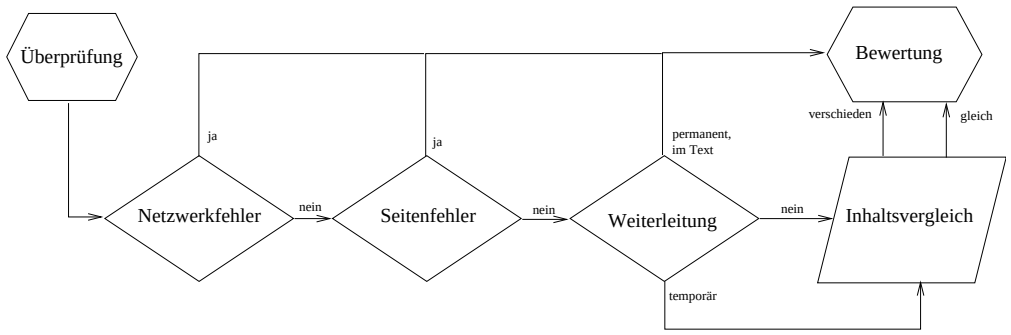


Abbildung 4.3: Arbeitsablauf von `URLChecker`

Wie kommt `URLChecker` nun zu der Bewertung der Seite? Zunächst wird die Verfügbarkeit des Inhaltes überprüft. Sollte diese aufgrund eines Netzwerkfehlers oder eines Seitenfehlers nicht gegeben sein ist die inhaltliche Bewertung nicht möglich, und der Status des Eintrages wird dem aufgetretenden Fehler entsprechend gesetzt. Sollte eine temporäre Weiterleitung auftreten, so wird die Überprüfung mit der Weiterleitungs-`URL` neu gestartet. Ansonsten wird der aktuelle Inhalt in Bezug auf eine Weiterleitung im Text analysiert und sollte diese Analyse negativ ausfallen, kann der aktuelle Inhalt mit der letzten Version verglichen werden. Der gesamte Arbeitsablauf ist in Abbildung 4.3 leicht vereinfacht dargestellt. Im Bezug auf den Inhaltsvergleich kann die Problematik aus Kapitel 2.3.2 wiederaufgenommen werden:

Wann entspricht der neue Inhalt noch dem alten?

Der in Kapitel 3.2.2 angesprochene Agent versucht die gefundenen Dokumente zum Beispiel anhand von Textklassifikation in eine gegebene Hierarchie einzuordnen. Für jede Stufe dieser Hierarchie ist dazu ein Klassifikator mit vielen Dokumenten aus dem jeweiligen Themengebiet trainiert worden. Hier zeigt sich schon das Problem: Im vorliegenden Falle kann nicht davon ausgegangen werden, das die überwachten Objekte in einer festen Hierarchie eingeordnet können. Eine Verwendung mehrerer (themenbasierter) Klassifikatoren die das Dokument bewerten scheidet aus. Eine mögliche Lösung dieses Problems ist die Verwendung eines Klassifikators, der die Dokumente nicht direkt bewertet, sondern die Beziehung zwischen zwei Dokumenten. Dieser Vergleich wird dann wie folgt realisiert (siehe auch Abbildung 4.4):

1. Berechne für jedes Maß die Distanz der beiden Dokumentversionen.
2. Bilde einen Vektor aus allen Ergebnissen[1].

[1] Jedes Distanzmaß hat eine eigene, konstante Position im Vektor

3. Klassifiziere den so erhaltenen Vektor durch den Klassifizierer und bewerte so die Inhalte als gleich, ähnlich oder ungleich. Die Entscheidung der Gleichheit oder Ähnlichkeit wird dabei durch einen vorgegebenen Schwellwert bestimmt.

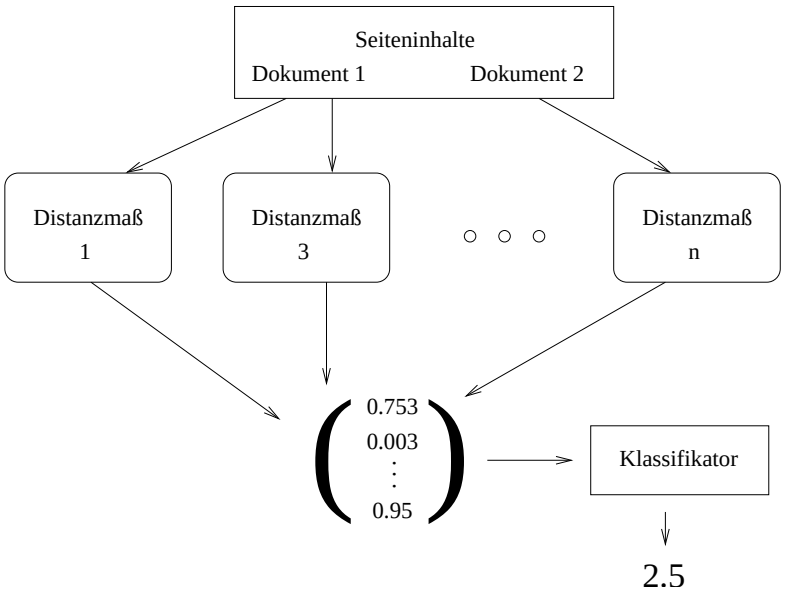


Abbildung 4.4: Ablauf des Inhaltsvergleiches

Man kann also sagen, das `URLChecker` die beiden Dokumente nicht direkt klassifiziert, sondern vielmehr das Ergebnis der Vergleiche. Der Vergleich bzw. die Klassifikation der Ähnlichkeitsvektoren abstrahiert von den konkreten Dokumenten und soll es ermöglichen, das Benutzerverhalten zu erkennen. Der verwendete Klassifikator — momentan eine Implementierung der Support Vector Machine (siehe Kapitel 3.3.1) —, lernt dabei im Prinzip die Gewichtung der einzelnen Distanzmaße anhand der Trainingsbeispiele, die durch vorgegebene Vergleichsbeipiele gegeben waren oder während des Betriebes durch Korrekturen an Entscheidungen von `URLChecker` entstanden sind.

4.5 Distanzmaße

Die Grundlage der Seitenbewertungen des `URLChecker`s sind die im folgenden vorgestellten Distanzmaße, deren Kombination zu einem Vergleichsvektor der verglichenen Dokumente führt. Dieser Vektor wird dann mit Hilfe des Klassifikators die Gesamtbewertung des Vergleiches ergeben.

Damit alle Distanzmaße vergleichbar sind, gelten folgende Konventionen für jedes Distanzmaß $sim(a, b)$:

1. Wertebereich: $0.0 \leq sim(a, b) \leq 1.0$
2. Größere Werte bedeuten größere Ähnlichkeit.

Diese Konventionen sind für den Lernvorgang anhand der ermittelten Vergleichsvektoren nötig. Ansonsten wäre es eventuell besser, einzelne Maße gerade nicht miteinander zu kombinieren, d.h. die Lernergebnisse wären bei Kombination aller Maße wesentlich schlechter als bei dem Einsatz einer Teilmenge der Maße. Dies läßt sich auf die mögliche unterschiedliche Bewertung der Ähnlichkeit zurückführen: Wo das eine Maß Werte nahe Null als große Ähnlichkeit definiert, macht ein anderes Maß es umgekehrt, oder ganz anders. Die resultierenden Vektoren können dann möglicherweise nicht mehr durch eine Hyperebene getrennt werden. Durch einen Fehler im Verlauf der Implementierung der Distanzmaße ist genau dieses Problem aufgetreten. Nachdem die fehlerhaften Maße angepasst wurden, entsprachen die Ergebnisse wieder den Erwartungen.

Die Distanzmaße lassen sich in vier grundlegende Klassen einteilen.

- URL-basierte Maße
  Die URL-basierten Maße berechnen ihre Bewertung anhand der `URL` der Seiten. Der Inhalt der Seiten wird nicht betrachtet. Die verschiedenen Ausprägungen dieser Klasse bewerten die `URL`s zum Beispiel anhand des Pfad- oder des Query-Anteils.

- Inhaltsbasierte Maße
  Im Gegensatz zu den URL-basierten Maßen beachten die inhaltsbasierten Maße ausschließlich den Seiteninhalt. Auch hier sind viele Variationen möglich, so kann ein Maß zum Beispiel den Volltext vergleichen, ein anderes Maß benutzt den euklidschen Winkel zwischen den Wortvektoren der Dokumente.

- Linkbasierte Maße
  Diese Maße machen sich die Link-Struktur der HTML-Dokumente zunutze und fällen ihre Entscheidung anhand der Übereinstimmungen zwischen den Links der beiden zu vergleichenden Seiten. Dabei kann die Reihenfolge der Links im Dokument beachtet werden, oder auch nicht. Außerdem unterscheiden sich die Maße durch den zur Bewertung benutzten Teil des Links (Linktext, LinkURL).

- Strukturbasierte Maße
  Diese Maße versuchen die Seiten anhand ihrer Struktur zu vergleichen. Da die Strukturanalyse von Seiten sehr schwierig ist, beschränken sich die hier vorgestellten Maße auf einfache Techniken, wie zum Beispiel dem Vergleich aller Überschriften in einer Seite.

4.5.1 URL-basierte Maße

Die URL-basierten Maße betrachten ausschließlich die URLs der Dokumente. Dies kann nützlich sein, wenn eine Seite im Inhalt geändert wurde, aber sich ihr Ort nur geringfügig geändert hat. Außerdem ist im Bezug auf die Gebrauchsart (siehe 2.1) zu sagen, daß sich der Seiteninhalt je nach Gebrauchsart der Seiten stark unterscheiden kann. Ein Beispiel ist hier z.B. das aktuelle Kinoprogramm, das meist unter der gleichen (oder ähnlichen) URL zu finden ist, dessen Inhalt sich aber stetig ändert.

Die Ähnlichkeit der Dokumente wird anhand mehrerer Komponenten in der URL berechnet. Die URLs werden zunächst in ihre Bestandteile (Protokoll, Servername, Pfad, Dateiname, Query, Anker, Benutzer (siehe 2.2.1.1)) zerlegt. Anschließend werden diese Komponenten unterschiedlich gewichtet und verknüpft.

4.5.1.1 URLMeasure

Dieses Maß betrachtet zunächst das Protokoll. Ist dieses nicht gleich, so wird 0.0 zurückgegeben, ansonsten berechnet sich die Ähnlichkeit anhand der Kombination der Ähnlichkeit von Servername, Dateiname und Pfad:

$$\begin{aligned} sim_{\mathrm{URL}}(a,\, b) &= 0.3 * \mathrm{HOST}(a, b) \\ &+ 0.3 * \mathrm{FILE}(a, b) \\ &+ 0.4 * 1.0 - \frac{tokenDistance\,(\mathrm{PATH}(a),\, \mathrm{PATH}(b))}{\max\left(\|\mathrm{PATH}(a)\|,\, \|\mathrm{PATH}(b)\|\right)} \end{aligned} \tag{4.1}$$

Sollten die zu vergleichenden Dokumente zum Beispiel auf unterschiedlichen Servern liegen, aber in einem ähnlichen Pfad, so wird die Ähnlichkeit mitunter größer sein, als zwischen zwei Dokumenten in stark unterschiedlichen Pfaden auf dem gleichen Server.

4.5.1.2 URLQueryMeasure

Das URLQueryMeasure arbeitet sehr ähnlich zum URLMeasure. Im Gegensatz zum URLMeasure wird hier aber zusätzlich zu den vorhergehenden Komponenten die Query-Komponente der Dokument-URLs berücksichtigt. Die Ähnlichkeit ergibt sich nach der folgenden Formel:

$$\begin{aligned} sim^{\mathrm{Query}}_{\mathrm{URL}}(a,\, b) &= 0.25 * \mathrm{HOST}(a, b) \\ &+ 0.25 * \mathrm{FILE}(a, b) \\ &+ 0.3 * 1.0 - \frac{tokenDistance\,(\mathrm{PATH}(a),\, \mathrm{PATH}(b))}{\max\left(\|\mathrm{PATH}(a)\|,\, \|\mathrm{PATH}(b)\|\right)} \end{aligned}$$

$$+ \quad 0.2 * 1.0 - \frac{tokenDistance\left(\mathrm{QUERY}(a),\, \mathrm{QUERY}(b)\right)}{\max\left(\left\|\mathrm{QUERY}(a)\right\|,\, \left\|\mathrm{QUERY}(b)\right\|\right)} \tag{4.2}$$

Durch die Berücksichtigung der Query-Komponente können in der `URL` übermittelte Benutzerdaten zur Bewertung der Ähnlichkeit genutzt werden.

4.5.1.3 URLQueryOnlyMeasure

Dieses Maß berechnet die Ähnlichkeit der `URL`s anhand der Query-Komponenten der `URL`s. Der Vergleich basiert auf der Editierdistanz zwischen den einzelnen Teilen der Query-Komponenten:

$$sim_{\mathrm{URL}}^{\mathrm{QueryOnly}}(a,\, b) \quad = \quad 1.0 - \frac{tokenDistance\left(\mathrm{QUERY}(a),\, \mathrm{QUERY}(b)\right)}{\max\left(\left\|\mathrm{QUERY}(a)\right\|,\, \left\|\mathrm{QUERY}(b)\right\|\right)} \tag{4.3}$$

Im Vergleich zu den anderen URL-Maßen hängt die Bewertung hier ausschließlich von der Query-Komponente der Dokument-`URL` ab. Andere Komponenten werden ignoriert.

4.5.2 Inhaltsbasierte Maße

Die inhaltsbasierten Distanzmaße vergleichen zwei Dokumente, wie schon durch den Namen zu vermuten war, anhand des Seiteninhaltes. Dieser Vergleich kann aber, wie gleich zu sehen, auf viele verschiedene Weisen erfolgen. Die verschiedenen Ansätze reagieren unterschiedlich auf Unterschiede zwischen den Dokumenten, so toleriert das RawTexMeasure zum Beispiel gar keine Änderungen, während sich die Wortpositionen im WordBagMeasure beliebig unterscheiden können.

4.5.2.1 SimpleTextMeasure und SimpleRawTextMeasure

Diese Maße sind die einfachsten, aber auch die schlechtesten aller inhaltsbasierten Maße. Die Texte werden einfach anhand des Volltextes verglichen, die Bewertung ist 1.0, wenn die Texte übereinstimmen, ansonsten 0.0. Das SimpleTextMeasure benutzt die „cooked“-Version des Seiteninhaltes, das SimpleRawTextMeasure den Rohtext.

Es ist leicht ersichtlich, daß schon kleinste Änderungen an einer Seite, wie z.B. die Beseitigung von Tippfehlern, oder auch nur ein zusätzliches Leerzeichen dazu führen, daß

die Versionen als unterschiedlich angesehen werden. Die Bewertungen können also sehr verwirrend für den Benutzer erscheinen, wenn nur diese Maße eingesetzt werden.

$$sim^{\text{Simple}}_{\text{Text}}(a,\, b) \;=\; \begin{cases} 1.0, \text{wenn } \text{TEXT}(a) = \text{TEXT}(b) \\ 0.0, \text{sonst} \end{cases} \tag{4.4}$$

$$sim^{\text{Simple}}_{\text{RawText}}(a,\, b) \;=\; \begin{cases} 1.0, \text{wenn } \text{DATA}(a) = \text{DATA}(b) \\ 0.0, \text{sonst} \end{cases} \tag{4.5}$$

4.5.2.2 Levensthein-Measures (Text, RawText)

Die Levensthein-Maße benutzen wie der Name schon vermuten lässt die Editierdistanz zwischen den beiden Seiteninhalten. Die Ermittlung dieser Distanz erfolgt wortbasiert anhand des „cooked"- bzw. des Rohtextes der Dokumente. Die Editierdistanz wird hierbei auf die Dokumentlänge (bzw. die maximale Editierdistanz) normalisiert, damit der gleiche Unterschied bei größeren Dokumenten nicht so stark gewichtet wird, wie bei kleineren Dokumenten.

$$sim^{\text{Levensthein}}_{\text{Text}}(a,\, b) \;=\; 1.0 - \frac{tokenDistance\left(\text{TEXT}(a),\, \text{TEXT}(b)\right)}{\max\left(\|\text{TEXT}(a)\|,\, \|\text{TEXT}(b)\|\right)} \tag{4.6}$$

$$sim^{\text{Levensthein}}_{\text{RawText}}(a,\, b) \;=\; 1.0 - \frac{tokenDistance\left(\text{DATA}(a),\, \text{DATA}(b)\right)}{\max\left(\|\text{DATA}(a)\|,\, \|\text{DATA}(b)\|\right)} \tag{4.7}$$

4.5.2.3 WordBagMeasure

Dieses Maß sieht die Dokumente als Wortmengen an. Für beide Dokumente wird jeweils die Menge aller darin befindlichen Worte gebildet, so daß Worte, Sätze oder Abschnitte innerhalb des Dokumentes beliebig verschoben werden können, ohne die Ähnlichkeit zu beeinflussen. Die Ähnlichkeit der Dokumente berechnet sich dann anhand der Schnittmenge der beiden Wortmengen bezogen auf die Gesamtmenge aller Worte. Dadurch werden größere Unterschiede bei großen Dokumenten abgeschwächt.

$$sim_{\text{WordBag}}(a,\, b) \;=\; \frac{\|\text{TEXT}(a) \cap \text{TEXT}(b)\|}{\|\text{TEXT}(a) \cup \text{TEXT}(b)\|} \tag{4.8}$$

4.5.2.4 TextDocumentvectorMeasures

Die Distanzmaße des TextDocumentvector-Typs verwenden zum Vergleich der Dokumente die Darstellung der Dokumente als Wortvektor. Zunächst werden die beiden Dokumente mit Hilfe eines Vektorisierers[2] in Wortvektoren umgewandelt. Dabei werden momentan

[2]Standard-Vektorisierer sind in `BotIShelly` verfügbar

alle Wörter aus beiden Dokumenten in den Wortvektor übernommen. Die Ähnlichkeit der beiden Dokumente entspricht dann bei dem EuklidDistanceTextDocumentvectorMeasure dem euklidschen Abstand der beiden Vektoren, das EuklidAngleTextDocumentvectorMeasure benutzt das Cosinusmaß, um die Ähnlichkeit zu bestimmen. Die zusätzlichen Formelteile dienen der Einhaltung der Distanzmaß-Konventionen.

$$sim^{\text{Angle}}_{\text{TextVector}}(a,\, b) \;=\; 1.0 - \left| \frac{\overrightarrow{\text{TEXT}(a)} \cdot \overrightarrow{\text{TEXT}(b)}}{\left\|\overrightarrow{\text{TEXT}(a)}\right\| * \left\|\overrightarrow{\text{TEXT}(b)}\right\|} \right| \tag{4.9}$$

$$sim^{\text{Distance}}_{\text{TextVector}}(a,\, b) \;=\; \exp\left(-\left\|\overrightarrow{\text{TEXT}(a)} - \overrightarrow{\text{TEXT}(b)}\right\|\right) \tag{4.10}$$

4.5.3 Linkbasierte Maße

Die linkbasierten Distanzmaße beruhen auf der Annahme, das sich die Ähnlichkeit zweier Dokumente durch die in ihnen enthaltenen Links beschreiben läßt. Da die linkbasierten Maße die Ähnlichkeit nur anhand der in den Dokumenten enthaltenen Links berechnen, sind diese Maße tolerant gegen inhaltliche Änderungen im Textkörper. Im Prinzip wird ein Unterteil der Struktur der Dokumente berücksichtigt, man kann diese Maße also zwischen die Klasse der inhaltsbasierten und die Klasse der Strukturmaße stellen.

Bei den linkbasierten Maßen gibt es zwei Arten:

- Maße, die die Link-URL benutzen
  Als Grundelement wird die URL des Links benutzt. Dieser Ansatz favorisiert den referentiellen Gebrauch von Links (siehe 2.1).
- Maße, die den Link-Text benutzen
  Als Grundelement wird der Text des Links benutzt. Dieser Ansatz favorisiert den attributiven Gebrauch von Links (siehe 2.1).

Für beide Arten gibt es (derzeit) drei Varianten, die die Gesamtmenge der Links in beiden Dokumenten unterschiedlich ordnet und verknüpft:

- Liste
  Hier wird die Reihenfolge der Links beachtet. Die Ähnlichkeit ist um so größer, je mehr Links an korrenspondierenden Positionen in den Dokumenten übereinstimmen.
- Menge (Bag)
  Für beide Dokumente werden jeweils alle Link-Elemente in einer Menge gesammelt. Anschließend werden diese Mengen miteinander verglichen. Die Ähnlichkeit

ist umso größer, je mehr Links in der Schnittmenge der beiden Link-Mengen vorhanden sind.

- Obermenge (Superset)
  Für beide Dokumente werden jeweils alle Link-Elemente in einer Menge gesammelt. Im Gegensatz zu der vorhergehenden Variante werden aber nur die Link-Elemente im zweiten Dokument berücksichtigt, die auch im ersten vorhanden sind, d.h. Links die im zweiten Dokument hinzugekommen sind werden vernachlässigt.

Die Berechnungsformeln der konkreten Link-Distanzmaße sind in Tabelle 4.2 zusammenfassend dargestellt. Zur Vereinfachung der Gleichungen werden die folgenden Hilfsfunktionen definiert:

$$
\begin{aligned}
\mathrm{LINKS}(D) &: \text{Liefert die Menge aller Links in } D. \\
\mathrm{LINKS}_{\mathrm{Text}}(D) &: \text{Liefert die Menge aller Link-Texte in } D. \\
\mathrm{LINKS}_{\mathrm{URL}}(D) &: \text{Liefert die Menge aller Link-URLs in } D. \\
\sigma_i^{\mathrm{URL}}(D) &: \text{Liefert den } i\text{-ten Link-URL aus Dokument } D. \\
\sigma_i^{\mathrm{TEXT}}(D) &: \text{Liefert den } i\text{-ten Linktext aus Dokument } D. \\
\theta(D) &= \|\mathrm{LINKS}(D)\|
\end{aligned}
$$

Sollte die Anzahl der Links in beiden Dokumenten 0 sein, dann wird die Ähnlichkeit mit 1.0 festgelegt. Im Falle der Obermengen-Variante gilt dies für $\theta(a)$.

4.5.4 Strukturbasierte Maße

In dieser Kategorie wird die Struktur der Dokumente analysiert und verglichen. Als Struktur ist hier nicht die natürlichsprachliche Struktur des Textes gemeint, sondern die HTML-Struktur der Dokumente. Dahinter verbirgt sich die Annahme, das sich die Ähnlichkeit zweier Dokumente durch die Ähnlichkeit ihrer Struktur ausdrücken lässt, d.h. zwei Seiten werden als ähnlich angesehen, wenn ihre Struktur ähnlich ist. Leider ist der umfassende Vergleich von Strukturen sehr aufwendig, so daß sich die in dieser Arbeit realisierten Maße auf sehr einfache Techniken beschränken.

4.5.4.1 LevensteinStructureMeasure

Dieses Distanzmaß wandelt die Dokumente in eine Folge von Tag-Bezeichnern um, alle anderen HTML-Elemente werden ignoriert. Diese Folgen werden anschließend mit Hilfe

Link-URL-Maße	Link-Text-Maße
URLLinkMeasure (4.11) $sim^{\text{URL}}_{\text{Link}}(a,\, b) = \dfrac{\sum_{i=0}^{\min(\theta(a),\theta(b))} \sigma_i^{\text{URL}}(a) = \sigma_i^{\text{URL}}(b)}{\max(\theta(a),\, \theta(b))}$	**TextLinkMeasure** (4.12) $sim^{\text{TEXT}}_{\text{Link}}(a,\, b) = \dfrac{\sum_{i=0}^{\min(\theta(a),\theta(b))} \sigma_i^{\text{TEXT}}(a) = \sigma_i^{\text{TEXT}}(b)}{\max(\theta(a),\, \theta(b))}$
URLLinkBagMeasure (4.13) $sim^{\text{URL}}_{\text{LinkBag}}(a,\, b) = \dfrac{\lVert \text{LINKS}_{\text{URL}}(a) \cap \text{LINKS}_{\text{URL}}(b) \rVert}{\lVert \text{LINKS}_{\text{URL}}(a) \cup \text{LINKS}_{\text{URL}}(b) \rVert}$	**TextLinkBagMeasure** (4.14) $sim^{\text{TEXT}}_{\text{LinkBag}}(a,\, b) = \dfrac{\lVert \text{LINKS}_{\text{URL}}(a) \cap \text{LINKS}_{\text{URL}}(b) \rVert}{\lVert \text{LINKS}_{\text{URL}}(a) \cup \text{LINKS}_{\text{URL}}(b) \rVert}$
TextLinkSupersetMeasure (4.15) $sim^{\text{TEXT}}_{\text{LinkSup}}(a,\, b) = \dfrac{\lVert \text{LINKS}_{\text{Text}}(a) \cap \text{LINKS}_{\text{Text}}(b) \rVert}{\theta(a)}$	**URLLinkSupersetMeasure** (4.16) $sim^{\text{URL}}_{\text{LinkSup}}(a,\, b) = \dfrac{\lVert \text{LINKS}_{\text{URL}}(a) \cap \text{LINKS}_{\text{URL}}(b) \rVert}{\theta(a)}$

Tabelle 4.2: Berechnungsformeln für die Link-Distanzmaße

der Editierdistanz verglichen. Das Ergebnis wird dann auf die maximale Größe der Editierdistanz (Länge des größeren Dokumentes) normiert, um größere Abstände bei großen Dokumenten zu erlauben.

$$sim^{\text{Levensthein}}_{\text{Structure}}(a,\, b) \quad = \quad 1.0 - \frac{tokenDistance\,(\text{TAGS}(a),\, \text{TAGS}(b))}{\max\left(\|\text{TAGS}(a)\|\,,\, \|\text{TAGS}(b)\|\right)} \tag{4.17}$$

4.5.4.2 HeadingMeasure

Das HeadingMeasure versucht anhand der Ähnlichkeiten der Überschriften die Ähnlichkeit der Dokumente zu messen. Es werden alle Überschriften (`H1-H5`-Tag), sowie überschriftenähnliche Gebilde (`BIG`-Tag, sehr große Schrift) aus den Dokumenten extrahiert. Die Berücksichtigung überschriftenähnlicher Gebilde ergibt sich aus den Überlegungen in Kapitel 2.2.2. Obwohl die Gebilde syntaktisch anders beschrieben sind, erscheinen sie dem Benutzer im Browser gleich. Über die extrahierten Elemente wird dann die Editierdistanz gebildet, wobei ein Element als eine Einheit behandelt wird. Die einzelnen Elemente werden also nicht weiter aufgespalten.

$$\begin{aligned} \phi\,(D) \quad &= \quad \left\{ t \in \text{TAGS}(D) \;\middle|\; \begin{array}{ll} & T = H1 \ldots H5,\, BIG \\ \vee & T = FONT \wedge fontsize(T) > 3 \end{array} \right\} \\ sim^{\text{Heading}}_{\text{Structure}}(a,\, b) \quad &= \quad 1.0 - \frac{tokenDistance\,(\phi\,(a),\, \phi\,(b))}{\max\left(\|\phi\,(a)\|\,,\, \|\phi\,(b)\|\right)} \end{aligned} \tag{4.18}$$

4.6 Benutzeroberfläche

Momentan präsentiert sich die Benutzeroberfläche des `URLChecker`s in einem Webbrowser. Der URLChecker selbst ist aber nicht auf diese Oberfäche angewiesen, d.h. es können gegebenenfalls auch andere Benutzeroberflächen benutzt werden. Außerdem wird durch diese getrennte Realisierung eine Anbindung an die Klassenbibliothek `BotIShelly` in Form eines Operators ermöglicht. Dieser Operator kann dann als Vergleichsoperator für verschiedene `URL`s benutzt werden. Angedacht ist zum Beispiel die Verwendung als Distanzmaß im PageTracker von N. Malzahn [Malzahn, 2003]

4.6.1 Das URLCheckerServlet

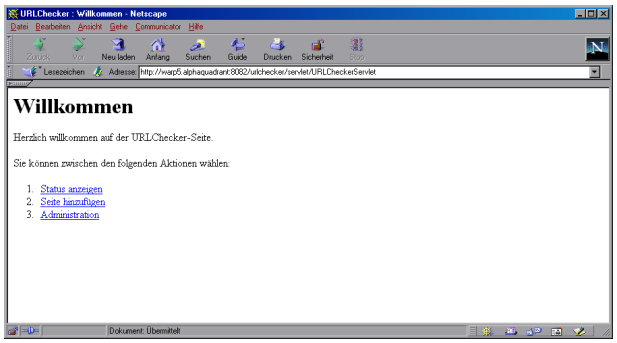


Abbildung 4.5: `URLChecker`-Servlet – Startseite

Zu Beginn wird man von der Hauptseite des `URLChecker`s (siehe Abbildung 4.5) empfangen. Hier stehen einem die grundlegenden Funktionen des `URLChecker`s zur Verfügung:

1. Anzeige der Statusübersicht (siehe Abbildung 4.6):
   Auf dieser Seite wird dem Benutzer der aktuelle Status aller `URL`s gezeigt, die der `URLChecker` in seiner Datenbank hat. Er kann sich die abgespeicherten Inhalte ansehen, Einträge löschen und den Status bestehender Seiten ändern.

2. Eingabe einer neuen `URL` die überwacht werden soll (siehe Abbildung 4.7):
   Hier kann der Benutzer eine neue `URL` zur Datenbank des `URLChecker`s hinzufügen.

3. Administrative Befehle (siehe Abbildung 4.8):
   Die administrativen Befehle umfassen Aktionen wie das Laden und Speichern der Konfiguration und der `URL`-Datenbank, sowie die Beendigung des Servlets.

4.6.1.1 Statusübersicht

In der Statusübersicht finden sich die Funktionen für den täglichen Gebrauch des `URL-Checkers`. Die Statusseite hat einen listenorientierten Aufbau, der die folgenden, in der Abbildung gekennzeichneten Hauptelemente besitzt:

1. Status-Eintrag für die `URL`
   Im Status-Eintrag der `URL` sind die allgemeinen Informationen des Eintrages vorhanden. Es wird der Name des Eintrages (die `URL`, mit der der Eintrag hinzugefügt wurde) sowie der aktuelle Status angezeigt (OK, nicht OK). Außerdem wird eine Statistik über die in dem Eintrag vorhandenen Elemente angezeigt.

   Der Benutzer kann hier die sofortige Überprüfung (3) oder die Entfernung des Eintrages aus der Datenbank (4) veranlassen. Die Überprüfung führt zu einem neuen Prüf-Eintrag im aktuellen Status-Eintrag. Sollte die Anzahl der Elemente die Höchstanzahl überschreiten, so werden alte Einträge automatisch gelöscht.

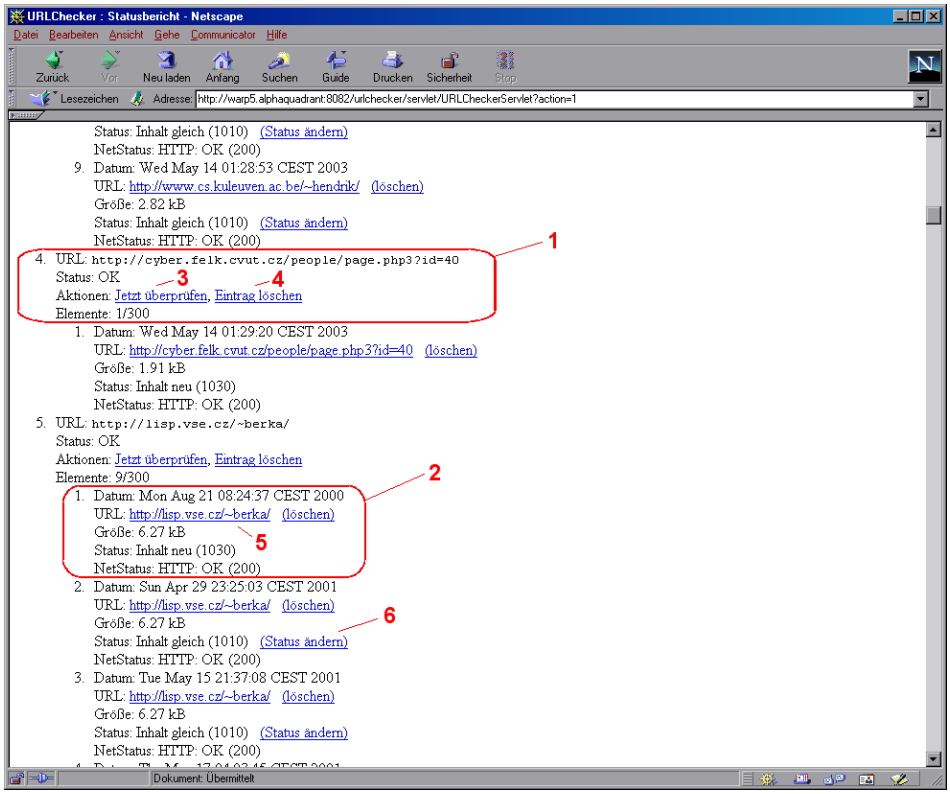


Abbildung 4.6: `URLChecker`-Servlet – Statusübersicht

2. Status-Eintrag für eine Überprüfung (Prüf-Eintrag)
   In diesem Eintrag kann der Benutzer die speziellen Daten für eine Überprüfung der `URL` sehen. Angezeigt werden das Datum und die `URL` (bei einer Weiterleitung die neue `URL`), sowie Größe der Datei und der Status zu dem der `URLChecker` gekommen ist. Außerdem wird der HTTP-Response-Code angezeigt.

   Ist die `URL` im Eintrag als Link angezeigt(5), so kann sich der Benutzer den Inhalt dieser Version anzeigen lassen, indem er diesem Link folgt. Sollte kein Inhalt verfügbar sein weil der Server z.B. nicht erreichbar war, ist die `URL` nur als Text vorhanden.

   Ist der Status des Eintrages „Inhalt gleich (1010)“, „Inhalt ähnlich (1020)“ oder „Inhalt verschieden (1040)“, so kann der Benutzer über die Funktion „Status ändern“(6) die Entscheidung des `URLChecker`s über diese Seite korrigieren, das Benutzerfeedback wird in diesem Falle ausgenutzt, um ein Trainingsbeispiel für die Lernphase zu erzeugen.

4.6.1.2 Eingabe einer neuen `URL`

Abbildung 4.7: `URLChecker`-Servlet – Eingabe einer neuen `URL`

Auf dieser Seite kann der Benutzer eine neue `URL` zu der `URLChecker`-Datenbank hinzufügen. Nach Eingabe und Übermittlung der `URL` wird diese sofort überprüft und in die Liste in der Statusübersicht eingefügt. Nach 5 Sekunden wird der Benutzer dann automatisch zur Statusübersicht weitergeleitet.

4.6.1.3 Administration

Alle Aktionen auf der Administrationsseite sind aus Sicherheitsgründen mit einem Passwort geschützt. Das richtige Passwort vorausgesetzt hat der Benutzer die folgenden Möglichkeiten:

1. Laden/Speichern der Datenbank
   Die `URL`-Datenbank wird vom Dateisystem geladen bzw. gespeichert. Im Falle des Ladevorgangs gehen alle Daten verloren, die bis zu diesem Zeitpunkt noch nicht abgespeichert waren!

2. Laden/Speichern der Konfiguration
   Der `URLChecker` liest bzw. schreibt seine Konfiguration und initialisiert sich entsprechend der neuen Konfiguration. Bei diesem Vorgang werden die Distanzmaße neu initialisiert und die `URL`-Datenbank neu gelesen. Es können also auch hier nicht gespeicherte Änderungen verloren gehen!

3. `URLChecker` beenden
   Dies führt zu einer Beendigung des Servlets und der virtuellen Maschine, in der das Servlet läuft.

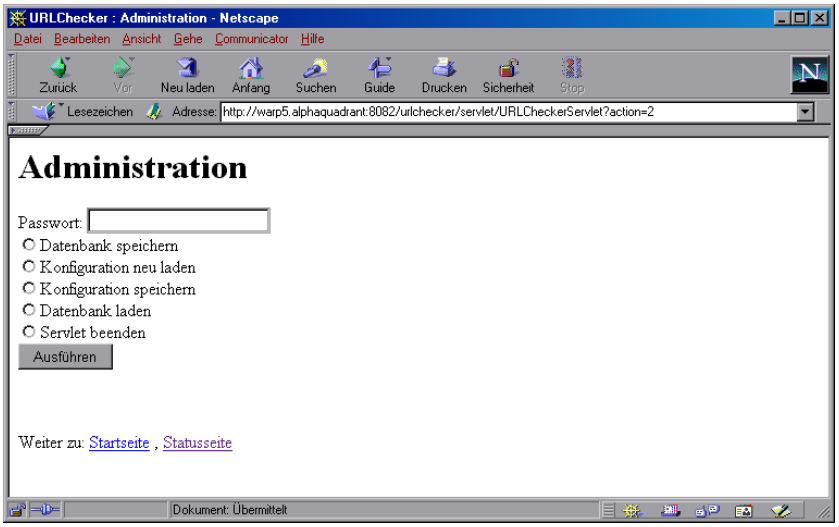


Abbildung 4.8: `URLChecker`-Servlet – Administration

4.6.2 Der URLCheckerLearner

Im Laufe der Zeit sammeln sich durch die Überprüfungen neue Vergleichsbeispiele an, die zur Anpassung des von `URLChecker` verwendeten Modells dienen können. Insbesondere die von dem Benutzer durchgeführten Statuskorrekturen liefern Informationen über Fehlentscheidungen und damit über Fehler im Modell. Der `URLChecker`Learner soll dem Benutzer bzw. Administrator dazu dienen, diese neuen Fakten in das Modell zu integrieren. Durch die – gegebenfalls manuelle – Bewertung der neuen Beispiele kann der Benutzer Einfluß auf die von `URLChecker` getroffenen Entscheidungen nehmen. Außerdem hat der Benutzer auch die Möglichkeit Beispiele zu entwerten, die seiner Meinung nach ungünstig sind. Diese Beispiele werden dann im Lernlauf nicht berücksichtigt. Bevor der Lernlauf gestartet wird, ist die Anpassung verschiedener Parameter und die Auswahl der genutzten Distanzmaße möglich.

4.6.2.1 Die Bewertungseite

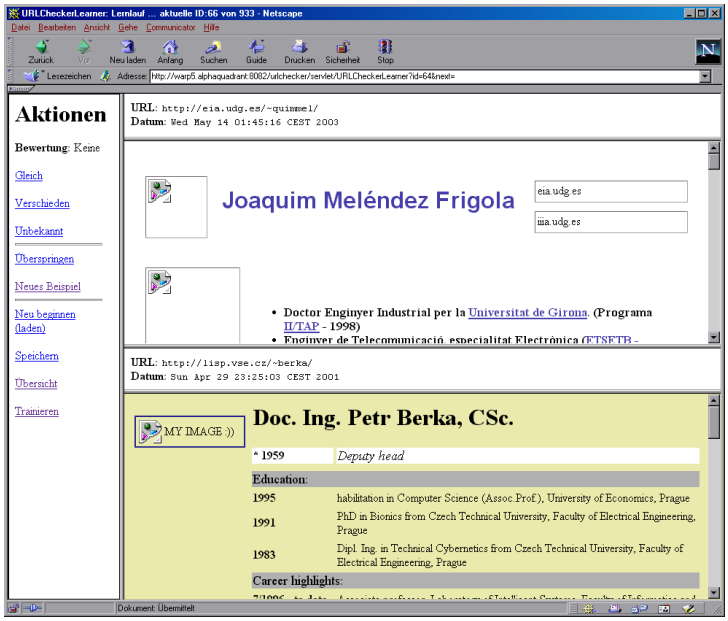


Abbildung 4.9: `URLCheckerLearner`-Servlet – Bewertungsseite

Die Bewertungsseite wird am Anfang des Lernlaufes automatisch mit dem ersten, noch nicht klassifizierten Beispiel der Lernmenge initialisiert. Von dieser Seite aus kann der Benutzer alle benötigten Aktionen ausführen und Entscheidungen zu den Beispielen treffen.

Die Seite ist als Frameset organisiert, auf der linken Seite befindet sich die Aktionsübersicht, die rechte Seite ist in zwei Frames für die Beispiele des Lernbeispieles aufgeteilt. Für die beiden Seiten des Beispieles wird jeweils die `URL` und das Datum, sowie der Inhalt angezeigt.

Die Aktionsübersicht ist in mehreren Bereichen aufgebaut. Im ersten Bereich sind die Aktionen vorhanden, die die Bewertung des aktuellen Lernbeispieles ändern (Gleich, Verschieden, Unbekannt). Der Benutzer hat außerdem die Möglichkeit, das aktuelle Beispiel zu überspringen oder ein neues Beispiel einzugeben (siehe Kapitel 4.6.2.2). Die Aktionen im unteren Teil beziehen sich auf den gesamten Lernlauf. Der Benutzer kann hier:

- Die Änderungen verwerfen (Neu beginnen)
- Die Änderungen sichern (Speichern)
- Zur Bewertungsübersicht springen (Übersicht)
- Die Trainingsparameter bestimmen und trainieren (Trainieren)

Wird das aktuelle Beispiel bewertet oder übersprungen, so wird der Lernlauf mit dem nächsten, noch nicht klassifizierten Beispiel fortgesetzt. Bei Fehlentscheidungen kann der Benutzer später über die Bewertungsübersicht eventuelle Korrekturen vornehmen.

4.6.2.2 Manuelle Beispieleingabe

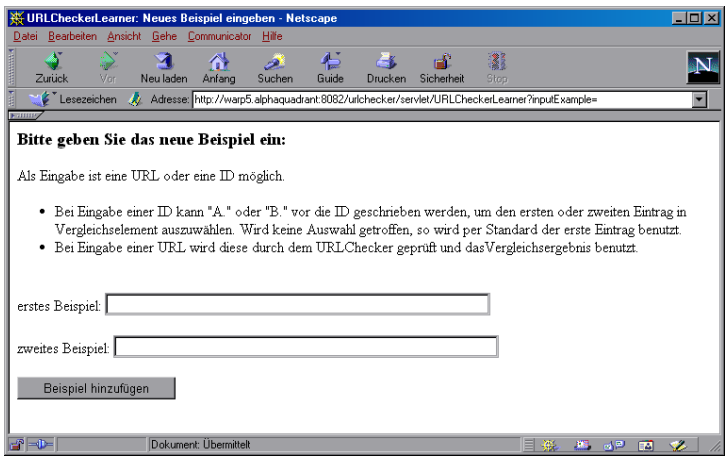


Abbildung 4.10: `URLCheckerLearner`-Servlet – Eingabe eines neuen Beispieles

Sollte der Benutzer ein – seiner Meinung nach – positives bzw. negatives Beispiel gefunden haben, so hat er auf der Seite für die manuelle Beispieleingabe die Möglichkeit,

dieses in die Lerndatenmenge aufzunehmen. Es ist sowohl die Eingabe von URLs möglich, sowie die Angabe einer Beispiel-ID eines schon vorhandenen Beispiels. Sollte die Eingabe einer URL verwendet werden, so wird diese automatisch durch den URLChecker überprüft und das daraus resultierende Element für das Beispiel genutzt.

4.6.2.3 Bewertungsübersicht

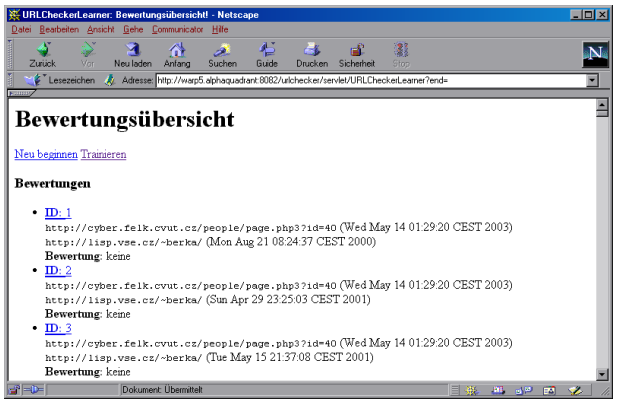


Abbildung 4.11: URLCheckerLearner-Servlet – Bewertungsübersicht

Die Bewertungsübersicht gibt dem Benutzer einen Überblick über die vorhandene Lernmenge. Zu jedem Beispiel werden URL und Datum, sowie die Bewertung (gleich, ungleich, keine) angezeigt. Von hier aus kann der Benutzer die Änderungen verwerfen (Neu beginnen), oder den Trainingslauf starten (Trainieren). Desweiteren ist die Übersicht gut dazu geeignet, zu einzelnen Beispielen zu springen, da jeder Eintrag durch einen Hyperlink zu der jeweiligen Bewertungsseite des Beispiels führt. Hier besteht also noch die Möglichkeit, eventuelle Korrekturen an den Entscheidungen vorzunehmen.

4.6.2.4 Vorbereiten des Trainings

Nachdem der Benutzer den Start des Trainingslaufs über die Bewertungsübersicht oder die Bewertungsseite ausgewählt hat, sind noch einige Lernparameter einzustellen, bevor der tatsächliche Trainingslauf beginnen kann (siehe Abbildung 4.12):

1. Kernel-Typ
   Hier kann die für den Trainingslauf gewünschte Kernfunktion der SVM (siehe Kapitel 3.3.6) ausgewählt werden. In der Evaluierung hat sich herausgestellt, das die polynomielle Kernfunktion sehr oft gute Ergebnisse liefert. Aber für die konkrete Beispielmenge kann natürlich eine andere Kernfunktion bessere Ergebnisse liefern.

2. Maximale Lernzeit
   Nach Ablauf dieser Zeit wird der Trainingslauf abgebrochen, sofern zu diesem Zeitpunkt noch kein Ergebnis vorliegt. Bei Einsatz der linearen Kernfunktion ist es zum Beispiel möglich, daß der Trainingslauf durch ungünstige oder fehlerhafte Daten sehr lange braucht, um zu einem Ergebnis zu kommen. Gegebenenfalls können dann anschließend einzelne Parameter für einen neuen Trainingslauf verändert werden.

3. Distanzmaße
   Der Benutzer hat hier die freie Wahl über alle im Trainingslauf zu berücksichtigenden Distanzmaße. So ist zum Beispiel der Ausschluß einzelner Maße oder auch Distanzmaßklassen möglich.

4. Transduktion
   Die verwendete Implementierung der SVM hat die Möglichkeit unbewertete Beipiele für den Trainingslauf zu benutzen, um den Suchraum für die Hyperebene „einzuschränken“ bzw. die unbewerteten Beispiele in der Optimierung zu berücksichtigen. Die Verwendung dieser Fähigkeit kann durch diese Option gesteuert werden.

Hat der Benutzer alle Parameter nach seinen Wünschen angepasst, so kann der Trainingslauf beginnen. Die Ausgaben der svm_{light} während der Lernphase werden dabei an den Benutzer übertragen. Ist die Lernphase erfolgreich abgeschlossen benutzt `URLChecker` ab diesem Zeitpunkt das neu gelernte Benutzermodell.

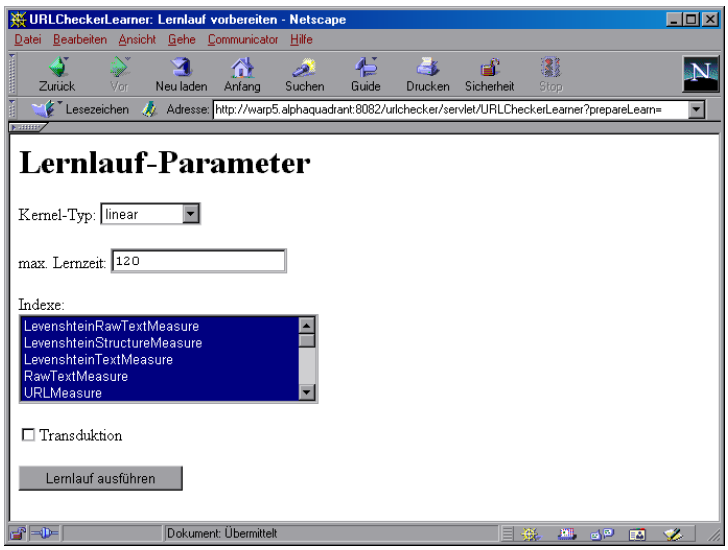


Abbildung 4.12: `URLCheckerLearner`-Servlet – Eingabe der Lernparameter

Kapitel 5

Auswertung

Dieses Kapitel widmet sich der Auswertung bzw. Evaluierung des `URLChecker`-Agenten. In den letzten Kapiteln wurde bereits deutlich, das `URLChecker` ein Modell des Benutzers benötigt, um die Seitenbewertungen an das Benutzerverhalten anzupassen. Zu diesem Zweck werden Benutzerentscheidungen benötigt, die in Form von positiven und negativen Beispielen vorhanden sein müssen. Zur Messung der Performanz des Agenten ist also eine Testmenge mit schon klassifizierten Beispielen notwendig. Doch wie kommt man an die Beispielmenge und deren Klassifikationen? Zwei Probleme, für die es verschiedene Lösungsstrategien gibt.

Eine Möglichkeit besteht in der manuellen Auswahl und Klassifikation der Seiten und Beispiele. Da die Menge der Beispiele jedoch groß ist, würde diese Arbeit sehr viel Zeit in Anspruch nehmen, wenn sie von nur einer Person erledigt werden sollte. Ein möglicher Ausweg aus dieser Situation wäre eine empirische Studie mit Hilfe von Testpersonen. Man motiviert viele Testpersonen, die dann eine Menge von Beispielen erzeugt, oder klassifiziert. Hier zeigen sich auch die Probleme dieser Vorgehensweise:

- Man muß zunächst einmal eine geeignet große Anzahl von Testpersonen finden, die die Studie mitmachen, um repräsentative Ergebnisse zu erhalten. Jeder der so etwas mal versucht hat wird wissen, das der Aufwand dafür sehr hoch ist.
- Wenn die Testpersonen die Beispiele nicht selbst erzeugen, sondern nur klassifizieren, so muß diese Beispielmenge erst einmal erzeugt werden, was ja auch schon ein Problem für sich darstellt.
- Angesichts der hohen Anzahl der Beispiele ist die Bewertung „von Hand“ trotzdem noch sehr aufwendig. Da sich viele Beispiele relativ ähnlich sehen, wird diese Aufgabe außerdem schnell als einseitig oder langweilig empfunden.

Aus diesen Gründen kommt diese Form der Evaluierung nur für Arbeiten in Frage, bei denen der Hauptteil der Arbeit auf die Studie selbst angelegt ist, was in dieser Arbeit nicht der Fall ist.

Eine andere Möglichkeit besteht in der synthetischen Generierung der Seiten selbst sowie der Beispiele und den dazugehörigen Klassifikationen. Um diese Art nutzen zu können ist aber meist schon im voraus einige Kenntnis über die „gewünschten“ Ergebnisse erforderlich. Schließlich kann man nicht *einfach so* Seiten – hier HTML-Seiten – erzeugen und aus diesen dann die Beispiele, sondern muß bei der Erzeugung mit dem Hintergrund von gezielten Prüfungen bestimmter Eigenschaften des Testobjektes (hier Agenten) vorgehen.
Eine Alternative zu diesen Methoden ist die automatische Generierung und Klassifikation von Beispielen, die auf schon gegebenen Daten basieren. Im Gegensatz zu der vorherigen Art werden hier nicht die den Beispielen zugrundeliegenden Seiten selbst erzeugt, sondern lediglich die „Komposition“ real existierender Seiten. Da die benutzten Seiten wirklich existieren ist man mit den Tests zumindest bei diesem Kriterium nah an der Realität. Die Zusammenstellung der Beispiele und deren Bewertung ist aber noch immer ein wichtiger, nicht vernachlässigender Punkt. Diese Vorgehensweise der semisynthetischen Generierung wird deshalb für die benutzten Beispielmengen genutzt.

Auf der anderen Seite stellt sich die Frage warum man denn überhaupt Tests durchführen muß. Es dürfte jedem klar sein, das eine Sache nichts nutzt, wenn sie ihrem Zweck nicht mehr oder weniger erfolgreich erfüllt. Mit anderen Worten nutzt einem der Agent nichts, wenn die Ergebnisse schlecht sind und damit auch der Nutzen gering ist. In Anlehnung an die Einleitung sei hier wieder der Fall des Administrators angeführt: Der verwendete Agent ist nutzlos, wenn er dem Administrator nicht einen großen Teil der Arbeit abnimmt, weil zum Beispiel für jede kleinste Änderung eine Meldung ausgelöst wird. Auf der anderen Seite sind zu wenige oder gar keine Meldungen bei Änderungen natürlich auch nicht gut. Der Administrator müßte wiederum eine manuelle Überprüfung vornehmen.
Letztendlich will der Entwickler auch sehen, ob der Agent die an ihn gestellten Erwartungen erfüllt, oder nicht. Man hat sich ja bei der Planung und Erstellung Ziele gesetzt und für diese gilt es in der Evaluation Ergebnisse zu finden.

Es stellt sich nun wiederum die Qualitätsfrage: *Was ist ein gutes Ergebnis?* Zu diesem Zweck wurden verschiedene Performanzmaße entwickelt, um die gewonnenen Ergebnisse mit anderen Experimenten vergleichen zu können. Einige populäre Performanzmaße, die auch zur Auswertung benutzt werden, werden in Kapitel 5.1 vorgestellt.
Kapitel 5.4 befasst sich mit der Auswahl der Beispiele aus der Menge der Testseiten und deren Bewertung, nachdem die Zusammensetzung der verwendeten Testseiten in Kapitel 5.3 erläutert wurde. Anschließend werden die Ergebnisse der erfolgten Testläufe ab Kapitel 5.5 vorgestellt.
Die in den dann folgenden Kapiteln geschilderten Ergebnisse beziehen sich auf die Problematiken in Bezug auf den Dokumentinhalt. Die in Kapitel 2.3.1 angesprochenen Fehler werden von `URLChecker` behandelt, bevor der Seiteninhaltsvergleich zustande kommt. Da diese Art von Fehlern protokollbasiert ist, besteht die Notwendigkeit der Anpassung

an das Benutzerverhalten nicht. Der Benutzer bewertet ja schließlich nur den Seiteninhalt. Desweiteren werden die inhaltlichen Fehler bezüglich der Weiterleitung automatisch behandelt. Für die protokollbasierten Weiterleitungen gilt die gerade angeführte Argumentation, die „Weiterleitung im Text“ wird ebenfalls vor der Inhaltsanalyse behandelt und führt gegebenenfalls zu einem anderen `URLChecker`-Status, durch den die inhaltliche Analyse verhindert wird.

5.1 Performanzmaße

Bevor die Auswertungen der Ergebnisse erfolgen, sollen die Performanzmaße zum besseren Verständnis in diesem Abschnitt erklärt werden. Zu den oft verwendeten Maßen gehören *Precision*, *Recall*, *Accuracy* und das *F-Measure*. Eine ausführliche Beschreibung von Precision und Recall ist in [Fuhr, 2000] zu finden. Für die Beschreibung der Performanzmaße wird Tabelle 5.1 zu Hilfe genommen.

Vorhersage	Realität	
	gleich	ungleich
gleich	p_+	n_+
ungleich	p_-	n_-

Tabelle 5.1: Kontingenztabelle für ein Klassifikationsproblem mit zwei Klassen.

5.1.1 Accuracy

Die *Accuracy* ist im maschinellen Lernen die gebräuchlichste Art zur Messung der Performanz eines Systems. Sie bezeichnet die Wahrscheinlichkeit, mit der ein zufällig aus der Verteilung der Beispiele gezogenes Beispiel richtig klassifiziert wird. In Bezug auf Tabelle 5.1 ergibt sich für die *Accuracy* die Formel:

$$Accuracy = \frac{p_+ + n_-}{p_+ + p_- + n_+ + n_-} \tag{5.1}$$

Oft wird anstatt der *Accuracy* auch die $Fehlerrate$ angegeben, die sich als Gegenwahrscheinlichkeit der *Accuracy* errechnet ($Accuracy = 1 - Fehlerrate$)
Durch den Einsatz eines Kostenmodells kann die *Accuracy* für manche Probleme praxisorientierter gemessen werden, wenn zum Beispiel die Kosten für eine Fehlentscheidung in den beiden Klassen unterschiedlich sind. So ist es für ein Diagnosesystem in der Medizin, das Patienten in die Klassen „Krebs“ und „kein Krebs“ einteilt wesentlich schlimmer,

einen Patienten mit Krebs als gesund anzusehen, als einen gesunden Patienten als krank zu bewerten. Durch die Verwendung eines Kostenmodells kann diese Nebenbedingung berücksichtigt werden.

5.1.2 Precision und Recall

Auch *Precision* und *Recall* können anhand von Tabelle 5.1 beschrieben werden. Die *Precision* drückt dabei die Wahrscheinlichkeit aus, das ein vom System als positiv eingestuftes Beispiel tatsächlich relevant ist, während der *Recall* die Wahrscheinlichkeit bezeichnet, daß ein tatsächlich relevantes Beispiel auch vom System erkannt wird (siehe auch [Raghavan et al., 1989]).

$$Precision = \frac{p_+}{p_+ + n_+} \qquad Recall = \frac{p_+}{p_+ + p_-} \tag{5.2}$$

Die Precision und der Recall sind im Regelfall nicht gleichzeitig optimierbar, d.h. für einen höheren Recall muß eine niederige Precision in Kauf genommen werden, und umgekehrt. Deshalb werden auch oft sogenannte Precision/Recall-Graphen betrachtet, bei denen der Recall gegen die Precision aufgetragen ist.

5.1.3 F-Measure

Das *F-Measure* ist als harmonischer Durchschnitt zwischen *Recall* und *Precision* definiert und geht auf das von Van Rijsbergen definierte *E*-Measure ([Rijsbergen, 1979]) zurück. Es ist ein einzelnes Maß für die Effektivität eines Systems, in dem Recall und Precision gleich gewichtet werden, wobei höhere Werte einer besseren Perfomanz entsprechen.

$$FMeasure = \frac{2 * Precision * Recall}{Precision + Recall} \tag{5.3}$$

$$= \frac{2p_+}{2p_+ + p_- + n_+} \tag{5.4}$$

5.2 Wayback-Machine

Zu Beginn der Evaluierung des Agenten stellte sich die Frage nach der Erzeugung der Testdaten: „Wie findet man in angemessener Zeit genügend Seiten, die sich im Zeitraum der Evaluierung hinreichend oft ändern?“
Das `WWW` ändert sich zwar stetig, jedoch ist es nicht vorhersehbar *wo* und *wann* Änderungen auftreten. Da für die Evaluierung nur ein begrenzter Zeitraum zur Verfügung stand, mußte nach einer Möglichkeit gesucht werden, die geforderten Seiten zu finden.

Hier kommt der Einsatz des Wayback-Archives ([Archive, Internet, 2001]) ins Spiel. Mit Hilfe der Internet-Wayback-Machine ist es möglich, in vergangene Zeiten des WWW zu „reisen“. Dieses Archiv speichert zu vielen Seiten vergangene Versionen, die direkt oder über eine Suchfunktion wieder abgerufen werden können. Also mußte man jetzt für die gewünschten Seiten „nur noch“ das Vorhandensein im Wayback-Archiv überprüfen. Ist die Seite vorhanden, so kann man einfach auf vergangene Versionen der Seite zugreifen. Zur Testdatengenerierung ist eine JAVA-Klasse entwickelt worden, die als Eingabeparameter eine URL enthält, für die alle geänderten Versionen heruntergeladen und nach einer Nachbearbeitung in die Datendank des URLCheckers aufgenommen wurden. Die Nachbearbeitung korrigierte dabei Eigenschaften, die von der Wayback-Machine abgeändert wurden und erzeugte eine „alte“ Version der Seite. So werden Wayback-Kommentare entfernt und die im Dokument enthaltenen Links korrigiert, um die Testergebnisse nicht zu verfälschen.[1]

Mit Hilfe der Wayback-Machine konnte so der Einsatz des Agenten über einen langen Zeitraum „simuliert“ werden.

5.3 Beschreibung der Testmenge

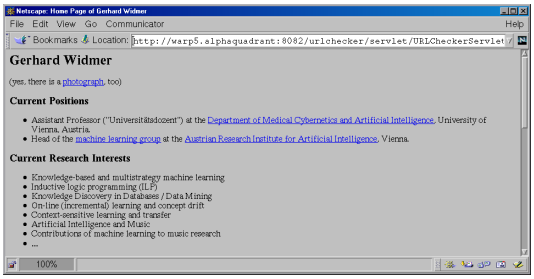


Abbildung 5.1: TestSet-1 – Homepage

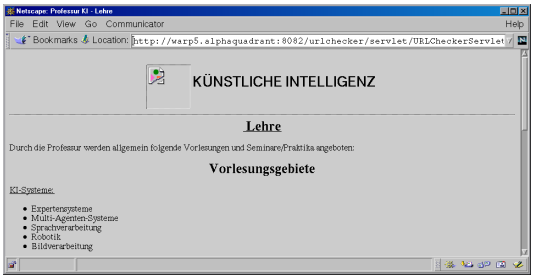


Abbildung 5.2: TestSet-1 – Kursseite

Die erste verwendete Testmenge (TestSet-1) setzt sich aus 31 unterschiedlichen URLs aus MlNet[2] zusammen und besteht zum Teil aus Homepages von Personen aus MlNet, sowie aus einigen Kursseiten dieser Personen (siehe Abbildung 5.1 und 5.2). Die verwendeten URLs für TestSet-1 sind beispielhaft in Tabelle 5.2 zu finden, die Liste der URLs für TestSet-2, sowie die ausführlichen Listen der verwendeten Vergleichselemente für beide Testmengen sind zusammen mehrere Seiten lang und hier aus diesem Grunde nicht abgebildet. Sie befinden sich zusammen mit der URL-Liste von TestSet-1 in Anhang A.

[1] Die Wayback-Machine lenkt z.B. alle Links auf sich selbst, um den zeitlichen Zusammenhang zu wahren. Im Bezug auf die Link-Maße ist dies natürlich nicht gewünscht.

[2] www.mlnet.org

http://ebusiness.hhl.de/staff/myra/
http://eia.udg.es/~quimmel/
http://lisi.insa-lyon.fr/~jfboulic/
http://lisp.vse.cz/~berka/
http://www-user.tu-chemnitz.de/~jzei/STAFF/dilger.html
http://www-user.tu-chemnitz.de/~jzei/zeidler.html
http://www.ai.univie.ac.at/~gerhard/
http://www.aifb.uni-karlsruhe.de/WBS/gst/
http://www.cs.bris.ac.uk/Teaching/Resources/COMS30106/
http://www.cs.bris.ac.uk/~flach/
http://www.cs.helsinki.fi/kurssit/
http://www.cs.kuleuven.ac.be/~hendrik/
http://www.cs.kuleuven.ac.be/~hendrik/ML/
http://www.cs.rhul.ac.uk/home/jhancock/lecnotes.html
http://www.csd.abdn.ac.uk/~pedwards/
http://www.csd.abdn.ac.uk/~pedwards/teaching/CS5505/slides.html
http://www.dsv.su.se/~henke/Welcome.html
http://www.few.eur.nl/few/people/bioch/
http://www.fi.muni.cz/~popel/
http://www.hhl.de/cxTP_contacts_1d.php?topic=study-ebusiness&id=49
http://www.iiia.csic.es/~mantaras
http://www.iiia.csic.es/~mantaras/
http://www.infc.ulst.ac.uk/~rayh/
http://www.infj.ulst.ac.uk/~cbcj23/files/teaching/lpai/lectures/
http://www.ipipan.waw.pl/ipi/staff/p.dembinski/
http://www.liacc.up.pt/~jgama/home_page_8.html
http://www.ncc.up.pt/~jgama/
http://www.pharmadm.com/people.shtml
http://www.rni.helsinki.fi/~htt/
http://www.soi.city.ac.uk/~eduardo/
http://www.tu-chemnitz.de/informatik/HomePages/KI/lehre.html

Tabelle 5.2: Liste der URLs von TestSet-1

Durch die Benutzung der Wayback-Machine als „Testdaten-Generator“ ist die Beipielmenge auf 199 Elemente (Überprüfungen) angestiegen, die sich von 1997 bis 2003 erstrecken. Aus diesen Grundelementen können dann durch das Szenario die Beipiele erstellt werden.

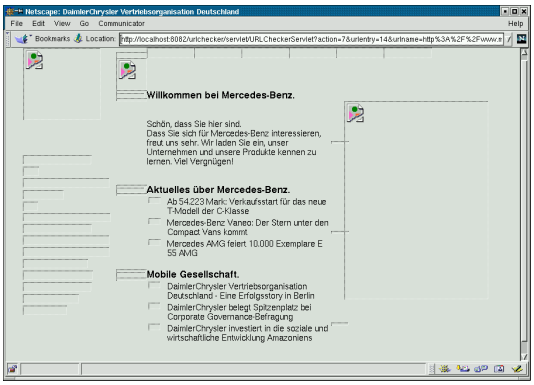


Abbildung 5.3: TestSet-2 – Autohersteller

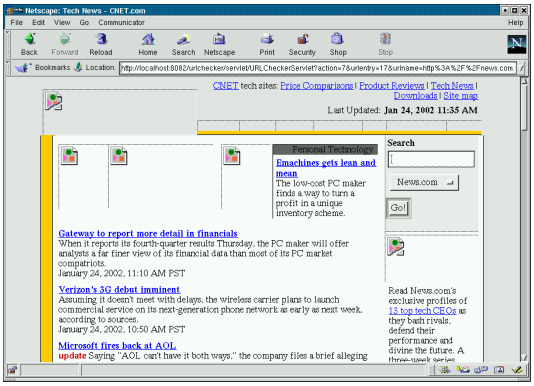


Abbildung 5.4: TestSet-2 – Newsseite

Die zweite Beispielmenge (TestSet-2) besteht im Gegensatz zur ersten Testmenge aus Webseiten vieler unterschiedlicher Bereiche, wie zum Beispiel Newsseiten oder Seiten von Fernsehsendern und Autoherstellern. Die stark variierenden Themen sollen den Agenten vor eine andere Situation als TestSet-1 stellen. TestSet-2 besteht aus 20 `URL`s und 249 Elementen (Überprüfungen), im Vergleich zu TestSet-1 wurden die `URL`s also häufiger überprüft. Die Versionen erstrecken sich auf einen Zeitraum von 1997 bis 2003, allerdings liegt der Schwerpunkt in den Jahren ab 2000.

5.4 Szenario

Da der `URLChecker` immer Seitenpaare miteinander vergleicht, wurde ein Szenario entwickelt, das nahe an der Realität in Bezug auf den Einsatz des Agenten liegt. Als positive Beispiele wurden die aufeinanderfolgenden Elemente eines Eintrages genutzt, also ist zum Beispiel für die `URL http://www.ai.univie.ac.at/~gerhard/` das Paar aus den Versionen vom 7.2.1997 und 6.2.1998 als positives Beispiel anzusehen. Als negative Beispiele wurden zwei Versionen mit unterschiedlichen `URL`s benutzt. Die Verwendung von Paaren mit kurzen zeitlichen Abständen als positive Beispiele entspricht so einem Benutzer, der Änderungen in einer Seite toleriert und verschiedene Seiten als unterschiedlich bewertet.
Durch die Kombination der Elemente ergaben sich für TestSet-1 1005 (168 positive und 837 negative) Beispiele als Grundmenge für die Evaluierung. Um den Einfluß der

ungleichen Verteilung von positiven und negativen Beispielen zu untersuchen, wurde jeweils ein gewichteter und ein ungewichteter Testlauf durchgeführt, d.h. im gewichteten Testlauf wurde die Anzahl der negativen Beispiele der Anzahl der positiven Beispiele angeglichen. Jeder Test wurde für die vier standardmäßig in der svm_{light} vorhandenen Kernmaße durchgeführt.
TestSet-2 hat 226 positive und 570 negative Beipiele (insgesamt 796). Die in Vergleich zu TestSet-1 andere Relation von positiven und negativen Beispielen ist durch die durchschnittlich höhere Anzahl von Überprüfungen pro `URL` in TestSet-2 zu erklären.

5.5 Ergebnisse

Wie in Kapitel 5.4 verdeutlicht, besteht die positive Testdatenmenge ausschließlich aus Beispielen deren `URL` gleich ist, entsprechend besteht die negative Testdatenmenge ausschließlich aus Paaren mit unterschiedlichen `URL`s. Aus diesem Grund ist zu erwarten, das die Tests mit optimalen Ergebnis ausfallen, wenn die URL-Maße – mit Ausnahme des URLQueryOnly-Maßes[3] – eingesetzt werden. Die experimentellen Ergebnisse bestätigen diese These und so wurden mehrere Distanzmaß-Kombinationen evaluiert, in denen die URL-Maße nicht eingesetzt wurden, um die Performanz der anderen Distanzmaße in Kombination zu messen.
In den folgenden Abschnitten werden nur vereinzelte Ausschnitte den Auswertungstabellen abgebildet oder aufbereitet, um bestimmte Gesichtspunkte bzw. Ergebnisse zu verdeutlichen. Alle Auswertungstabellen mit den gemittelten Performanzwerten der Testläufe sind in Anhang B einsehbar. Wenn im folgenden von einem *Lauf* gesprochen wird, ist damit der Test eines Distanzmaßes bzw. einer Distanzmaßkombination gemeint und sollte nicht mit dem Lauf im Anhang verwechselt werden, der für die einzelnen Durchgänge der Kreuzvalidierung steht.
Für alle Tests wurde eine Kreuzvalidierung mit fünf Teilen durchgeführt, um die Ergebnisse nicht durch eine ungünstig gewählte Testmenge zu verfälschen. Dabei wurden die positiven und negativen Beispiele jeweils durch zufälliges Ziehen in fünf Teile geteilt, um Lern- bzw. Testmengen zu vermeiden, die gar kein oder nur wenige positive Beispiele enthalten. Die im Anhang abgebildeten Tabellen enthalten dann die über die erfolgreichen Läufe gemittelten Performanzwerte, die durch den Einsatz der Mikrobewertung (siehe [Fuhr, 2000]) ermittelt wurden.
Die Tabellen 5.3 und 5.4 geben Vergleichswerte der Performanzmaße für die Klassifikatoren *Random* und *Default* wieder. *Random* wählt dabei für ein gegebenes Beispiel zufällig eine Klasse aus, *Default* wählt die Klasse mit den meisten Beispielen. Die Accuracy des

[3] Da das URLQueryOnly-Maß ausschließlich den Query-Teil der `URL` betrachtet und in der Testmenge nur eine `URL`s mit Query-Komponente vorkam, ist das Ergebnis der Lernhase immer ein Default-Klassifizierer.

Random-Klassifikators sollte von den Distanzmaßen nicht unterschritten werden (ansonsten könnte man besser das Ergebnis auswürfeln). Da sich die Werte in Bezug auf die Testmengen lediglich in der Accuracy unterscheiden, wurde auf separate Tabellen verzichtet. Der in Klammern angebene Wert der Accuracy bezieht sich auf TestSet-2.

Klassifikator	Precision	Recall	Accuracy	F-Measure
Random	0.5	0.5	0.5	0.5
Default	0.5	0.5	0.5	0.5

Tabelle 5.3: Performanzmaße anderer Klassifikatoren (gewichtet)

Klassifikator	Precision	Recall	Accuracy	F-Measure
Random	0.5	0.5	0.5	0.5
Default	N/V	0.0	0.833(0.716)	0.0

Tabelle 5.4: Performanzmaße anderer Klassifikatoren (ungewichtet)

5.5.1 Gewichtung

Wie in Kapitel 5.4 beschrieben, wurden die Tests jeweils gewichtet und ungewichtet durchgeführt. Im allgemeinen weichen die Werte der einzelnen Läufe nur geringfügig voneinander ab. Leichte Abweichungen der Accuracy sind einleuchtend, da die Anzahl der negativen Beipiele in der Berechnung dieses Performanzmaßes eingeht und genau diese Anzahl ja durch die Gewichtung geändert wurde.
Bei Einsatz der linearen Kernfunktion sind die Unterschiede teilweise erheblich, so hat das TextLinkMeasure angewendet auf die gewichtete Beipielmenge bei TestSet-1 durchschnittlich nur eine Accuracy von 0.542, welche als schlecht zu bewerten ist. Im Gegensatz dazu ist die Accuracy im ungewichteten Fall 0.970, was einen recht guten Wert darstellt. Durch die größere Anzahl negativer Beispiele wird die optimale Hyperebene so weit verschoben, das Bewertungsfehler des Distanzmaßes ausgeglichen werden können.
Auf der anderen Seite kann aber auch das genaue Gegenteil eintreten. Für TestSet-2 und URLLink-Maß ist z.B. der Accuracy-Wert des gewichteten Laufes 0.923, im ungewichteten Fall ist sie nur 0.871. Der Unterschied scheint nicht besonders groß zu sein, wenn man aber die Wahrscheinlichkeit, das ein positives Beispiel auch als ein solches erkannt wird (Recall) betrachtet, erkennt man den großen Unterschied: 0.845 (gewichtet) zu 0.544 (ungewichtet).
Im Regelfall sind diese Unterschiede aber auf unzureichende Distanzmaße zurückzuführen. Wenn wiedersprüchliche Werte geliefert werden, müssen diese zwangsläufig aufgelöst werden, indem Fehler zugelassen werden. Hier wird sich der Vorteil der kombinierten Maße bemerkbar machen.

5.5.2 Kernfunktionen

Im großen und ganzen sind die Ergebnisse der Läufe bei Einsatz der polynomiellen und radialen Kernfunktion vergleichbar. Die Unterschiede ergeben sich bei Betrachtung der lineraren und der sigmoiden Kernfunktion. Wie schon eben angedeutet ist bei Einsatz der linearen Kernfunktion gerade im gewichteten Fall bei einigen Einzelmaßen kein Ergebnis gefunden worden. Desweiteren ist zu beobachten, das es im Gegensatz zur polynomiellen und radialen Kernfunktion öfter Ergebnisse gibt, bei denen ein Default-Klassifikator herauskommt (z.B. TestSet-1, ungew., LevenstheinTextMeasure). Die Bewertungen dieser Distanzmaße sind anscheinend oft nicht linear separierbar, so daß die Benutzung anderer Kernfunktionen nötig ist.
Bei Verwendung der sigmoiden Kernfunktion ergeben sich zum größten Teil keine Ergebnisse für die Distanzmaßkombinationen und die URL-Maße. Von dem Einsatz der sigmoiden Kernfunktion ist bei Verwendung dieser Maße abzuraten. Wenn andere Einzelmaße verwendet werden liefert der Testlauf gute Ergebnisse, wobei mehr Ergebnisse mit fehlerhaft klassifizierten negativen Beipielen zu beobachten sind, als bei der Verwendung der anderen Kernfunktionen.

5.5.3 Distanzmaße

Nachdem in den vorgehenden Abschnitten auf den Einfluß der Gewichtung, sowie der Kernfunktion auf die Testergebnisse eingegangen wurde, soll hier die in der Einleitung gestellte Frage beantwortet werden: „Führt die Kombination mehrerer Distanzmaße zu einer Verbesserung der Ergebnisse?“

In den verschiedenen Testläufen gab es oft einzelne Distanzmaße,die bessere Performanzwerte als die kombinierten Maße hatten. So liefert das TextLinkBag-Maß für TestSet-1 oft eine bessere Accuracy als die kombinierten Distanzmaße (z.B. 0.97 vs. 0.95). Allerdings kehrt sich diese Beobachtung für TestSet-2 um. Hier erzielt das kombinierte Distanzmaß im Regelfall eine bessere Accuracy (z.B. 0.91 vs. 0.94). In einem Fall ergab der Lernlauf für dieses Einzelmaß sogar einen Default-Klassifikator, der die Seiten immer als unterschiedlich klassifizierte. Solche Ausnahmen kamen im Falle der kombinierten Verwendung „aller“ Maße[4] nie vor. Hat der Lernlauf zu einem Ergebnis geführt, waren die Performanzwerte immer gut.

Wenn man alle Einzelmaße betrachtet, kann man auch große Unterschiede erkennen. Es gibt Distanzmaße, wie z.B. die SimpleText-Maße deren Performanz nahezu immer schlecht ist. Diese schlechte Performanz ist aber auch schon in der Entwicklung erwartet

[4] URL-Maße ausgeschlossen

worden und soll nur aufzeigen, daß es auch ungeeignete Möglichkeiten gibt, bestimmte Änderungen zu erkennen (Genau diese Art wird aber von Webspidern eingesetzt). Für eine Beispielmenge mit einem ganz bestimmten Aufbau werden auch diese Maße gute Ergebnisse liefern, allerdings ist solch eine Beispielmenge in der Realität kaum zu erwarten. Ein gutes Beispiel für die mitunter starke Abhängigkeit der Einzelmaße von

Beispielmenge	Precision	Recall	Accuracy
TestSet-1 (gewichtet)	1.0	0.89	0.94
TestSet-1 (ungewichtet)	0.89	0.88	0.93
TestSet-2 (gewichtet)	0.56	0.97	0.60
TestSet-2 (ungewichtet)	N/V	0.0	0.72

Tabelle 5.5:
Gemittelte Performanzwerte des HeadingMeasure für verschiedene Beispielmengen

der Beispielmenge ist das Heading-Maß. Bei alleiniger Anwendung dieses Maßes ergeben sich für TestSet-1 gute Werte von Precision, Recall und Accuracy. Bei Anwendung auf TestSet-2 ist im ungewichteten Fall ein Default-Klassifikator und im gewichteten Fall ein Klassifikator mit sehr schlechter Precision herausgekommen (siehe Tabelle 5.5).

Bei den kombinierten Maßen gibt es ebenfalls bessere und schlechtere Kombinationen. Für die verwendeten Beispielmengen bildet die Gruppe der Strukturmaße das Schlußlicht, wobei Werte von 1.0 (Precision), 0.87 (Recall) und 0.95 (Accuracy) den Durchschnitt bilden, sofern man die sigmoide Kernfunktion außer Betracht läßt. Damit liegt diese Kombination aber schon im oberen Mittelfeld aller Distanzmaße und Maßkombinationen.

Distanzmaßkombination	Precision	Recall	Accuracy
nur strukturbasiert	1.0	0.89	0.95
nur linkbasiert	1.0	0.93	0.96
nur textbasiert	1.0	0.91	0.95
ohne URL-Maße	1.0	0.91	0.95
TextLinkSupersetMeasure	0.99	0.95	0.97
TextLinkBagMeasure	1.0	0.95	0.97
SimpleRawTextMeasure	1.0	0.23	0.62
EuklidDistancTextDocumentvectorMeasure	0.98	0.45	0.72

Tabelle 5.6: Performanzwerte verschiedener Distanzmaßkombinationen (TestSet-1, polynomielle Kernfunktion, gewichteter Testlauf)

Bessere Werte erzielen die Kombinationen der textbasierten sowie linkbasierten Maße. Die Performanzwerte der einzelnen Distanzmaßkombinationen sind in Tabelle 5.6 exemplarisch für den gewichteten Testlauf über TestSet-1 bei Verwendung der Kernfunktion „polynomiell“ abgebildet. Außerdem sind die Performanzwerte der besten sowie schlechten Einzelmaße zum Vergleich angegeben. In anderen Testläufen ändert sich die Rangfolge leicht, die Performanzwerte der Distanzmaßkombinationen bewegen sich aber immer in ähnlichen Bereichen, wobei auch Fälle auftreten, in denen die Maßkombinationen die besten Werte aufweisen. Tabelle 5.6 bildet somit einen Ausschnitt aus der Gesamttabelle für diese Parameterkombination, die gesamte Tabelle sowie die Tabellen anderer Lernläufe können in Anhang B nachgeschlagen werden.

Zusammenfassend kann die am Anfang gestellte Frage in Bezug auf eine einzelne Beispielmenge zwar nicht mit einem klaren „Ja“ beantwortet werden, da die kombinierten Maße nicht immer die besten Performanzwerte liefern. Allerdings ist es auch nicht immer das gleiche Einzelmaß, das die besten bzw. bessere Ergebnisse liefert und außerdem ist der „Verlust“ durch die Verwendung der Distanzmaßkombination nur sehr gering. Da in der Realität verschiedenste Beispielmengen zu erwarten sind, sollte man die Verwendung der kombinierten Distanzmaße der Verwendung von Einzelmaßen auf jeden Fall vorziehen. Bestimmte Distanzmaße mögen in Einzelfällen eine bessere Performanz liefern, allerdings stellen sich dabei auch folgende Fragen:

- Wie erkennt man das richtige Distanzmaß für die konkrete Beispielmenge?
  Eine automatische Bewertung aller Einzelmaße und anschließende Selektion wird in der Praxis zu aufwendig sein. Die Testläufe in dieser Arbeit haben jeweils mehrere Stunden benötigt, wohingegen das Training einer Distanzmaßkombination maximal mehrere Minuten, im Regelfall nur wenige Sekunden dauerte. Sollte sich die Anzahl der Distanzmaße erhöhen, steigt die benötigte Zeit linear mit der Anzahl der neuen Distanzmaße an, da die Lernläufe für jedes neue Einzelmaß durchgeführt werden müßten.
- Wie ist die Performanz, wenn sich die Beispielmenge ändert?
  Es kann sein, das das zuvor sehr gute Maß dann schlechte Ergebnisse liefert, mit der Folge einer neuen Evaluierung. Die Distanzmaßkombinationen sind anscheinend robuster als die Einzelmaße.
- Erfolgt eventuell eine Überanpassung des Einzelmaßes an die Beispielmenge?
  Da das beste aller Einzelmaße gewählt wird, ist die Gefahr einer Überanpassung des gelernten Modells an die Beispielmenge sehr hoch. Das gelernte Modell generalisiert nicht mehr von der gegebenen Beispielmenge, und somit ist es vermutlich nicht mehr auf neue Fälle übertragbar. Die Kombination verschiedener Distanzmaße steuert diesem Effekt entgegen.

Kapitel 6

Zusammenfassung und Ausblick

Der Agent `URLChecker` ist als ein Ergebnis dieser Arbeit entstanden, der mit Hilfe des maschinellen Lernens versucht, seine Entscheidungen besser als andere schon vorhandene Lösungen zu treffen. Gerade die in Kapitel 2.2.2 angesprochenen Probleme der Bewertung des Seiteninhaltes auch in Bezug auf die verwendete Gebrauchsart machen den Einsatz des maschinellen Lernens erforderlich, um diese Aufgabe zu lösen. Da die Themen der von `URLChecker` überwachten Seiten stark variieren können, konnte der „normale“ Ansatz der Textklassifikation nicht zur alleinigen Bewertung der Seite benutzt werden. Also wurde die Verwendung von Seitenpaaren als Vergleichselemente genutzt, um anschließend nicht die Bewertung der Beispiele, sondern die Bewertungen der Ähnlichkeitsmaße zu lernen. Durch den Einsatz verschiedener Distanzmaße, die unterschiedliche Aspekte im Dokument bewerten wurde versucht, die Bewertung flexibel an das Benutzerverhalten anpassen zu können.
Die derzeit vorhandenen Alternativen umgehen die Problematiken bezüglich des Seiteninhaltes einfach, indem sie diesen nicht berücksichtigen. Das ist natürlich ein nicht befriedigender Zustand.

In der Evaluierung wurde der Einfluß unterschiedlicher Lernparameter und die Verwendung unterschiedlicher Distanzmaße untersucht. Neben der Verwendung einzelner Maße wurde auch der Einsatz von Distanzmaßkombinationen betrachtet und es hat sich herausgestellt, daß der Einsatz der Kombinationen zwar nicht immer die besten Ergebnisse lieferte, dafür aber immer eine gute Performanz zu beobachten war. Die Performanz der Einzelmaße variierte je nach Parameter oder Beispielmenge unterschiedlich stark. Nie war *ein einziges* Einzelmaß *immer* besser als die kombinierten Maße. Abhängig von der Beispielmenge und den verwendeten Lernparametern lieferten unterschiedliche Einzelmaße oder auch kombinierte Maße die besten Performanzwerte.

Damit lassen sich die in der Einleitung gestellten Fragen beantworten:

- Ist die Verwendung von Seitenpaaren als Grundlage geeignet, um aus Seitenbewertungen zu lernen?

Die Ergebnisse der Evaluierung zeigen, daß das Lernen über den Ähnlichkeiten der Beispiele erfolgreich durchgeführt werden kann.

- Ist der Einsatz mehrerer Distanzmaße in Kombination sinnvoll?
  Führt die Kombination mehrerer Distanzmaße zu einer Verbesserung der Ergebnisse?
  Die Verwendung mehrerer Distanzmaße führt nicht immer zu den besten Ergebnissen, dafür aber stets zu guten Ergebnissen. Die Distanzmaßkombinationen sind aus diesem Grunde „robuster“ als Einzelmaße. Der Einsatz mehrerer Distanzmaße ist also auf jeden Fall sinnvoll, auch wenn sie nicht immer das optimale Ergebnis erzielen.

Natürlich ist die Vorgehensweise von `URLChecker` nicht allgemeingültig. Es gibt noch Probleme, die derzeit nicht gelöst sind und Wünsche, deren Lösung schön wäre:

- gemischte Gebrauchsarten
  Das Lernen eines Benutzermodells geht davon aus, das dieser Benutzer immer nach den gleichen Kriterien entscheidet. Was ist aber, wenn der Benutzer für ein Dokument sowohl attributive als auch referentielle Gesichtspunkte verwendet? Da das Dokument als eine Einheit angesehen wird, ist eine Unterscheidung von Dokumentteilen nicht möglich.
  Selbst die Verwendung unterschiedlicher Gebrauchsarten innerhalb der Beispielmenge kann zu Problemen führen. Wenn der Benutzer die Hälfte der Dokumente unter referentiellen und die andere Hälfte unter attributiven Gesichtspunkten betrachtet, so werden einige Distanzmaße die eine Hälfte der Dokumente priorisieren, andere Distanzmaße die andere. Ein gutes Lernergebnis ist dann aufgrund der Wiedersprüche in der Beispielmenge nicht zu erwarten. Vielleicht könnte die Beispielmenge untersucht und dann unterteilt werden, um die Lernergebnisse zu verbessern.

- Verwendung anderer Inhaltsträger
  Die Verwendung von Bildern, Flash-Animationen und JAVA-Skript steigt immer weiter an. Zur Zeit werden Inhalte dieser Art nicht vom Agenten unterstützt. Durch die Erweiterung der Agentenbibliothek `BotIShelly`, sowie die Entwicklung zusätzlicher Maße könnten diese Gesichtspunkte in die Bewertung von `URLChecker` aufgenommen werden. Im übrigen würde so die Flexibilität in Bezug auf unterschiedliche Benutzerverhalten verbessert werden. Sollte ein Distanzmaß nur schlechte Werte liefern, so wird es durch den Lernvorgang nur mit einem geringen Gewicht berücksichtigt. Die Ergebnisse werden sich also durch zusätzliche Maße nicht wesentlich verschlechtern.

- Mehrbenutzerunterstützung
  `URLChecker` ist momentan darauf ausgelegt, eine Datenbank und ein Modell zu

benutzen. Eine Erweiterung für den Mehrbenutzerbetrieb, durch die z.B. alle Benutzer die Möglichkeit haben, Beispiele anderer Benutzer zu verwenden könnte vielleicht die Ergebnisse verbessern und würde den Aufwand des einzelnen Benutzers verringern.

- Benutzung durch `BotIShelly`
  `URLChecker` kann durch die Verwendung des prototypisch implementierten Operators durch Agenten genutzt werden, die die Agentenbibliothek `BotIShelly` nutzen. Der Einsatz von `URLChecker` als Distanzmaß im Pagetracker-Agenten könnte dessen Suchergebnisse noch verbessern, die Integration und Auswertung der Ergebnisse ist derzeit aber noch nicht realisiert worden.

Literaturverzeichnis

[Archive, Internet, 2001] Archive, I. und Internet, A. (2001). Wayback Machine. `http://www.archive.org/`.

[Armstrong et al., 1995] Armstrong, R., Freitag, D., Joachims, T., und Mitchell, T. (1995). WebWatcher: A Learning Apprentice for the World Wide Web. In *AAAI Spring Symposium on Information Gathering*, Seiten 6–12.

[Banken et al., 2000] Banken, M., Fischbach, C., Geppert, O., Hövener, M., Jägersküpper, J., Kaschlun, V., Malzahn, N., Masloch, A., Mentel, U., Podvoiskaia, M., Schröter, N., Joachims, T., und Klinkenberg, R. (2000). BotIshelly – Bibliothek zur Erstellung von Agenten für die Suche im Internet. `http://www-ai.cs.uni-dortmund.de/LEHRE/PG/PG343`.

[Barz, 2002] Barz, W. (2002). Bibliographie zum Thema Intentionalität. `http://www.fu-berlin.de/philosophie/varia/Intentionalitaet/Barz%20-%20Bibliographie%20Intentionalitaet.pdf`.

[Berners-Lee, 1996] Berners-Lee, T. (1996). The Myth of Names and Addresses. `http://www.w3.org/DesignIssues/NameMyth.html`.

[Berners-Lee, D.Conolly, 1995] Berners-Lee, T. und D.Conolly (1995). *RFC 1866: Hypertext Markup Language*.

[Berners-Lee et al., 1998] Berners-Lee, T., Fielding, R., U.C.Irvine, und Masinter, L. (1998). Uniform Resource Identifiers (URI): Generic Syntax. `http://www.ietf.org/rfc/rfc2396.txt`.

[Bordihn, 2000] Bordihn, C. (2000). Ein lernender Agent zur kooperativen Erstellung und Verwaltung von themenorientierten Dokumentdatenbanken aus dem WWW. Diplomarbeit, Fachbereich Informatik, Universität Dortmund.

[Broder et al., 2000] Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., und Wiener, J. (2000). Graph structure in the web. In *9th WWW Conference*.

[Burges, 1998] Burges, C. J. C. (1998). A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167.

[Cohen, 1995] Cohen, W. W. (1995). Text categorization and relational learning. In Prieditis, A. und Russell, S. J., Hrsg., *Proceedings of ICML-95, 12th International Conference on Machine Learning*, Seiten 124–132, Lake Tahoe, US. Morgan Kaufmann Publishers, San Francisco, US.

[Donnellan, 1966] Donnellan, K. (1966). Reference and Definite Descriptions. *Philosophical Review*, 75:281–304.

[Fielding et al., 1999] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., und Berners-Lee, T. (1999). *RFC 2616: Hypertext Transfer Protocol – HTTP/1.1.* `ftp://ftp.isi.edu/in-notes/rfc2616.txt`.

[Forman, 2003] Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.*, 3:1289–1305.

[Fuhr, 2000] Fuhr, N. (2000). Information Retrieval — Skriptum zur Vorlesung im WS 00/01. `http://www.is.informatik.uni-duisburg.de/teaching/dortmund/lectures/ir_ws00-01/irskall.pdf`.

[Hirschberg, 1975] Hirschberg, D. S. (1975). A linear space algorithm for computing maximal common subsequences. *Communications of the ACM*, 18(6):341–343.

[Joachims et al., 1997] Joachims, T., Freitag, D., und Mitchell, T. M. (1997). Web Watcher: A Tour Guide for the World Wide Web. In *IJCAI (1)*, Seiten 770–777.

[Levensthein, 1965] Levensthein, V. I. (1965). Binary codes capable of correcting spurious insertions and deletions of ones. *Problems of Information Transmission*, 1:8–17.

[Malzahn, 2003] Malzahn, N. (2003). PageTracker - ein Agent zum Wiederauffinden von Webseiten mit intelligenten Suchanfragen. Diplomarbeit, Fachbereich Informatik, Universität Dortmund.

[N.Vapnik, 1995] N.Vapnik, V. (1995). *The Nature of Statistical Learning Theory.* Springer-Verlag New York, Inc., New York, USA.

[Raghavan et al., 1989] Raghavan, V., Bollmann, P., und Jung, G. S. (1989). A critical investigation of recall and precision as measures of retrieval system performance. *ACM Trans. Inf. Syst.*, 7(3):205–229.

[Rijsbergen, 1979] Rijsbergen, C. V. (1979). *Information Retrieval.* Butterworth, London, 2. Auflage.

[Salton, 1991] Salton, G. (1991). Developments in Automatic Text Retrieval. *Science*, 253:974–979.

[Salton, McGill, 1983] Salton, G. und McGill, M. (1983). *Introduction to modern information retrieval*. McGraw–Hill computer science series. McGraw–Hill, New York.

[The NCSA Mosaic Team, 1998] The NCSA Mosaic Team (1998). A Beginner's Guide to URLs. `http://archive.ncsa.uiuc.edu/demoweb/url-primer.html`.

[Wang, Hu, 2002] Wang, Y. und Hu, J. (2002). Detecting tables in html documents.

[Zeidler, 1996] Zeidler, E., Hrsg. (1996). *Teubner Taschenbuch der Mathematik*, Jgg. 1. B.G. Teubner Stuttgart.

Anhang A

Testdatenmenge

A.1 Testdatensatz 1

A.2 URLs

http://ebusiness.hhl.de/staff/myra/
http://eia.udg.es/~quimmel/
http://lisi.insa-lyon.fr/~jfboulic/
http://lisp.vse.cz/~berka/
http://www-user.tu-chemnitz.de/~jzei/STAFF/dilger.html
http://www-user.tu-chemnitz.de/~jzei/zeidler.html
http://www.ai.univie.ac.at/~gerhard/
http://www.aifb.uni-karlsruhe.de/WBS/gst/
http://www.cs.bris.ac.uk/Teaching/Resources/COMS30106/
http://www.cs.bris.ac.uk/~flach/
http://www.cs.helsinki.fi/kurssit/
http://www.cs.kuleuven.ac.be/~hendrik/
http://www.cs.kuleuven.ac.be/~hendrik/ML/
http://www.cs.rhul.ac.uk/home/jhancock/lecnotes.html
http://www.csd.abdn.ac.uk/~pedwards/
http://www.csd.abdn.ac.uk/~pedwards/teaching/CS5505/slides.html
http://www.dsv.su.se/~henke/Welcome.html

Fortsetzung auf nächster Seite

Fortsetzung von letzter Seite

http://www.few.eur.nl/few/people/bioch/
http://www.fi.muni.cz/~popel/
http://www.hhl.de/cxTP_contacts_1d.php?topic=study-ebusiness&id=49
http://www.iiia.csic.es/~mantaras
http://www.iiia.csic.es/~mantaras/
http://www.infc.ulst.ac.uk/~rayh/
http://www.infj.ulst.ac.uk/~cbcj23/files/teaching/lpai/lectures/
http://www.ipipan.waw.pl/ipi/staff/p.dembinski/
http://www.liacc.up.pt/~jgama/home_page_8.html
http://www.ncc.up.pt/~jgama/
http://www.pharmadm.com/people.shtml
http://www.rni.helsinki.fi/~htt/
http://www.soi.city.ac.uk/~eduardo/
http://www.tu-chemnitz.de/informatik/HomePages/KI/lehre.html

Tabelle A.1: URLs TestSet-1

A.3 Elemente

http://ebusiness.hhl.de/staff/myra/			
	30.04.2001, 14:50:05	09.08.2001, 02:56:57	22.08.2001, 07:33:26
	08.10.2001, 22:11:04	22.12.2001, 02:24:28	05.06.2002, 06:00:39
http://eia.udg.es/~quimmel/			
	24.04.2001, 14:17:47	24.07.2001, 00:00:23	25.10.2001, 18:42:52
	21.12.2001, 11:19:25	14.05.2003, 01:45:16	
http://lisi.insa-lyon.fr/~jfboulic/			
	24.04.2001, 13:50:52	15.05.2001, 23:20:33	05.10.2001, 23:39:45
	18.12.2001, 04:23:34	01.02.2002, 19:30:12	28.05.2002, 19:14:47
	14.05.2003, 01:31:01		
http://lisp.vse.cz/~berka/			
	21.08.2000, 08:24:37	29.04.2001, 23:25:03	15.05.2001, 21:37:08
	17.05.2001, 04:03:45	20.05.2001, 02:56:38	21.07.2001, 00:20:22

Fortsetzung auf nächster Seite

Fortsetzung von letzter Seite

30.10.2001, 02:13:46	04.02.2002, 15:12:50	15.04.2003, 19:24:34
`http://www-user.tu-chemnitz.de/~jzei/STAFF/dilger.html`		
11.06.2001, 03:00:39	31.10.2001, 21:01:52	08.12.2001, 02:43:39
15.04.2003, 19:14:30		
`http://www-user.tu-chemnitz.de/~jzei/zeidler.html`		
29.04.2001, 08:50:36	03.01.2002, 08:59:23	03.02.2002, 14:14:44
14.05.2003, 01:33:14		
`http://www.ai.univie.ac.at/~gerhard/`		
07.02.1997, 18:51:22	06.02.1998, 23:42:42	02.05.1998, 21:39:53
24.08.2000, 00:22:36	24.04.2001, 16:26:59	13.06.2001, 08:38:29
23.07.2001, 18:03:05	30.10.2001, 02:34:43	02.12.2001, 06:10:42
05.02.2002, 00:32:35	14.05.2003, 01:14:27	
`http://www.aifb.uni-karlsruhe.de/WBS/gst/`		
17.12.2000, 23:43:00	30.04.2001, 01:12:18	04.06.2001, 00:04:20
02.11.2001, 07:41:04	04.06.2002, 00:10:49	14.05.2003, 01:32:25
`http://www.cs.bris.ac.uk/Teaching/Resources/COMS30106/`		
05.10.1999, 06:11:20	25.09.2000, 09:21:15	30.04.2001, 01:58:14
06.06.2001, 22:30:24	12.08.2001, 05:45:01	24.12.2001, 20:08:29
05.06.2002, 05:50:15	14.05.2003, 01:38:21	
`http://www.cs.bris.ac.uk/~flach/`		
22.01.1998, 03:21:55	03.02.1999, 03:03:27	06.10.1999, 14:44:34
24.08.2000, 00:22:32	04.04.2001, 02:05:04	17.05.2001, 00:44:56
20.07.2001, 02:00:42	20.09.2001, 20:57:51	29.10.2001, 23:46:10
17.12.2001, 15:01:43	02.02.2002, 21:42:19	22.05.2002, 20:34:03
05.06.2002, 19:20:47	14.05.2003, 01:39:15	
`http://www.cs.helsinki.fi/kurssit/`		
21.10.1997, 18:22:18	22.01.1998, 07:11:00	02.09.2000, 11:20:59
29.01.2001, 10:03:00	17.02.2001, 05:46:01	13.04.2001, 05:21:12
08.06.2001, 10:15:31	30.10.2001, 08:24:35	02.02.2002, 10:34:49
02.06.2002, 20:51:14	15.04.2003, 19:19:12	
`http://www.cs.kuleuven.ac.be/~hendrik/`		
30.07.1997, 06:53:05	22.01.1998, 21:58:41	10.02.1999, 05:59:46
04.10.1999, 00:37:09	24.08.2000, 00:22:32	29.10.2001, 13:12:32
02.12.2001, 06:47:13	06.02.2002, 03:41:34	14.05.2003, 01:28:53

Fortsetzung auf nächster Seite

Fortsetzung von letzter Seite

`http://www.cs.kuleuven.ac.be/~hendrik/ML/`		
30.04.2001, 04:57:16	03.06.2001, 13:57:23	20.10.2001, 03:45:55
02.12.2001, 00:33:47	12.02.2002, 08:41:09	15.04.2003, 19:07:44
`http://www.cs.rhul.ac.uk/home/jhancock/lecnotes.html`		
12.06.2002, 09:52:30	04.06.2003, 00:08:43	
`http://www.csd.abdn.ac.uk/~pedwards/`		
01.12.1998, 20:39:21	18.02.1999, 08:03:28	31.08.2000, 08:53:01
24.01.2001, 05:21:00	04.04.2001, 07:23:26	29.10.2001, 19:27:46
04.02.2002, 15:08:51	22.05.2002, 21:14:51	15.04.2003, 19:16:08
`http://www.csd.abdn.ac.uk/~pedwards/teaching/CS5505/slides.html`		
11.10.2000, 00:19:23	05.01.2001, 05:46:00	08.04.2001, 03:53:52
25.12.2001, 07:55:24	19.02.2002, 09:53:28	15.04.2003, 19:15:50
`http://www.dsv.su.se/~henke/Welcome.html`		
13.01.1997, 00:40:47	05.12.1998, 05:14:20	20.04.1999, 06:24:47
29.04.1999, 20:32:41	05.11.1999, 17:25:49	14.04.2000, 03:20:45
15.06.2000, 00:00:35	07.02.2002, 02:12:39	15.04.2003, 19:17:51
`http://www.few.eur.nl/few/people/bioch/`		
03.12.1998, 15:52:11	09.05.2001, 08:32:46	24.07.2001, 11:33:13
10.01.2002, 11:51:28	12.06.2002, 06:17:34	15.04.2003, 19:21:03
`http://www.fi.muni.cz/~popel/`		
20.05.2001, 03:51:33	11.06.2001, 01:29:45	23.07.2001, 01:14:14
04.11.2001, 00:49:04	21.12.2001, 11:38:42	02.02.2002, 03:17:32
14.05.2003, 01:30:29		
`http://www.hhl.de/cxTP_contacts_1d.php?topic=study-ebusiness&id=49`		
15.04.2003, 19:01:51		
`http://www.iiia.csic.es/~mantaras`		
17.01.1999, 01:28:29	02.02.2002, 00:38:15	
`http://www.iiia.csic.es/~mantaras/`		
14.05.2003, 01:41:41		
`http://www.infc.ulst.ac.uk/~rayh/`		
02.10.1999, 18:29:49	24.08.2000, 00:22:28	24.04.2001, 14:44:23
19.05.2001, 20:27:38	09.06.2001, 08:22:27	24.07.2001, 02:19:25
30.10.2001, 02:07:33	18.12.2001, 03:40:51	04.02.2002, 08:31:09
26.05.2002, 11:02:51	04.06.2003, 00:04:01	

Fortsetzung auf nächster Seite

Fortsetzung von letzter Seite

`http://www.infj.ulst.ac.uk/~cbcj23/files/teaching/lpai/lectures/`		
10.12.2001, 11:19:10	05.06.2002, 19:44:13	04.06.2003, 00:03:48
`http://www.ipipan.waw.pl/ipi/staff/p.dembinski/`		
23.02.2002, 06:35:35	15.04.2003, 19:18:40	
`http://www.liacc.up.pt/~jgama/home_page_8.html`		
03.08.2001, 20:53:37	06.12.2001, 04:37:55	04.02.2002, 22:52:32
09.06.2002, 15:23:04	15.04.2003, 19:18:52	
`http://www.ncc.up.pt/~jgama/`		
24.05.1997, 05:43:07	05.12.1998, 13:34:16	03.02.1999, 23:08:54
05.10.1999, 03:18:03	30.08.2000, 15:05:55	04.04.2001, 05:00:26
21.07.2001, 10:59:54	06.10.2001, 15:31:24	06.10.2001, 15:31:34
04.06.2002, 03:22:23	04.06.2003, 00:07:30	
`http://www.pharmadm.com/people.shtml`		
14.12.2001, 16:39:51	04.02.2002, 15:31:38	04.06.2003, 00:08:41
`http://www.rni.helsinki.fi/~htt/`		
17.08.2000, 04:31:16	20.09.2001, 03:11:40	28.11.2001, 13:47:53
04.02.2002, 00:42:21	22.05.2002, 20:13:32	15.04.2003, 19:20:20
`http://www.soi.city.ac.uk/~eduardo/`		
29.08.2001, 18:17:55	26.10.2001, 00:13:23	05.12.2001, 16:29:28
05.02.2002, 07:17:20	22.05.2002, 23:36:22	27.05.2002, 21:39:46
14.05.2003, 01:41:06		
`http://www.tu-chemnitz.de/informatik/HomePages/KI/lehre.html`		
09.10.1999, 09:50:17	13.11.1999, 00:45:52	10.03.2000, 19:08:11
11.10.2000, 18:43:59	23.07.2001, 00:38:10	31.10.2001, 02:11:28
07.03.2002, 00:33:26	04.06.2003, 00:03:14	

Tabelle A.2: Elemente TestSet-1

A.4 Testdatensatz 2

A.5 URLs

http://news.com.com/

http://www-ai.cs.uni-dortmund.de/LEHRE/VORLESUNGEN/GVPR/1998WS/STUDIS/uebungsblaetter.html

http://www-vetpharm.unizh.ch/

http://www.1aeuropreis.de

http://www.knurzkurier.de/

http://www.lycos.de/

http://www.meine-erste-homepage.com/

http://www.mercedes-benz.de/

http://www.n-tv.de/

http://www.nur-ein-euro.de/

http://www.opel.de/

http://www.partypaket.de/

http://www.prosieben.de/home/

http://www.racheshop.com/

http://www.radio-essen.de/

http://www.sat1.de/

http://www.simpsonsweb.de/

http://www.strokes.de/

http://www.virtuelle-apotheke.de/

http://www.weltbild.de/

Tabelle A.3: URLs TestSet-2

A.6 Elemente

http://news.com.com/		
24.01.2002, 19:35:32	07.08.2002, 17:09:12	09.08.2002, 08:08:12
12.08.2002, 07:04:50	12.08.2002, 20:49:24	13.08.2002, 07:42:37
15.08.2002, 08:07:14	20.08.2002, 08:16:09	22.08.2002, 08:12:47
23.08.2002, 07:54:02	25.08.2002, 07:46:32	26.08.2002, 07:13:07
27.08.2002, 07:55:57	29.08.2002, 08:51:23	31.08.2002, 07:27:59
01.09.2002, 07:36:55	02.09.2002, 07:38:07	03.09.2002, 07:32:09
05.09.2002, 07:35:59	08.09.2002, 07:45:39	09.09.2002, 07:37:29
09.09.2002, 08:05:38	10.09.2002, 07:44:19	14.09.2002, 02:59:21
14.09.2002, 09:07:41	25.09.2002, 08:37:19	03.11.2003, 23:42:12
http://www-ai.cs.uni-dortmund.de/LEHRE/VORLESUNGEN/GVPR/1998WS/STUDIS/uebungsblaetter.html		
08.07.2000, 15:07:11	15.12.2000, 02:00:00	22.03.2003, 15:09:14
04.11.2003, 00:17:29		
http://www-vetpharm.unizh.ch/		
31.01.1997, 03:52:55	25.01.1999, 09:28:28	02.12.2001, 17:39:10
18.01.2002, 11:57:56	23.05.2002, 20:44:20	28.05.2002, 18:04:39
05.06.2002, 08:38:26	25.07.2002, 14:35:11	02.08.2002, 13:39:21
06.08.2002, 21:35:30	21.11.2002, 15:17:08	04.02.2003, 15:21:49
14.02.2003, 23:02:42	04.11.2003, 00:23:25	
http://www.1aeuropreis.de		
28.01.2003, 21:20:19		
http://www.knurzkurier.de/		
07.12.2002, 13:48:32	04.11.2003, 00:17:47	
http://www.lycos.de/		
11.12.1997, 19:08:38	01.12.1998, 06:41:12	12.12.1998, 03:07:05
25.01.1999, 10:09:48	08.02.1999, 01:46:06	25.02.1999, 13:01:21
21.04.1999, 04:17:04	27.04.1999, 23:44:44	07.10.1999, 03:58:34
01.03.2000, 17:33:51	01.03.2000, 18:41:19	02.03.2000, 09:26:21
04.03.2000, 01:26:31	08.04.2000, 16:52:23	10.05.2000, 09:32:00
10.05.2000, 18:33:30	11.05.2000, 00:12:28	11.05.2000, 00:39:18
11.05.2000, 13:10:52	12.05.2000, 01:29:21	20.05.2000, 00:03:37

Fortsetzung auf nächster Seite

Fortsetzung von letzter Seite

03.06.2000, 09:17:37	19.06.2000, 18:45:34	19.06.2000, 21:03:07
20.06.2000, 00:32:03	20.06.2000, 15:00:47	21.06.2000, 16:13:08
21.06.2000, 16:41:05	22.06.2000, 04:55:22	06.07.2000, 22:43:20
04.11.2003, 00:30:53		
`http://www.meine-erste-homepage.com/`		
15.09.2000, 15:33:21	01.02.2001, 08:53:00	31.03.2001, 03:26:11
17.05.2001, 02:12:25	22.07.2001, 01:00:56	24.09.2001, 01:12:50
27.11.2001, 10:16:15	21.01.2002, 21:54:36	23.01.2003, 22:28:40
04.11.2003, 00:25:50		
`http://www.mercedes-benz.de/`		
02.12.1998, 03:50:53	27.04.1999, 23:47:15	29.02.2000, 15:19:42
10.05.2000, 00:52:17	12.05.2000, 01:39:14	20.05.2000, 09:42:42
19.06.2000, 09:11:01	18.10.2000, 22:44:16	09.11.2000, 03:13:00
21.11.2000, 09:39:00	06.12.2000, 01:06:00	18.01.2001, 20:24:00
01.03.2001, 05:02:04	04.04.2001, 23:23:08	18.04.2001, 15:44:43
15.05.2001, 20:39:27	20.05.2001, 13:00:35	29.05.2001, 20:41:21
28.06.2001, 08:02:52	28.06.2001, 08:23:48	11.07.2001, 22:37:09
05.06.2002, 17:02:53	02.08.2002, 07:02:44	27.09.2002, 16:57:50
29.09.2002, 18:11:06	26.11.2002, 19:24:43	05.02.2003, 10:34:36
03.11.2003, 23:31:57		
`http://www.n-tv.de/`		
02.12.1998, 18:22:15	19.10.2000, 07:14:17	13.09.2001, 09:56:59
17.09.2001, 02:25:54	17.09.2001, 02:26:03	19.09.2001, 07:14:24
26.09.2001, 07:31:00	26.09.2001, 19:47:47	27.09.2001, 18:33:25
29.09.2001, 19:04:55	01.10.2001, 17:45:09	01.10.2001, 19:37:29
02.10.2001, 19:53:34	02.10.2001, 20:34:19	03.10.2001, 20:51:05
04.10.2001, 21:07:08	05.10.2001, 19:38:24	06.10.2001, 20:13:41
06.10.2001, 21:40:10	07.10.2001, 08:27:09	08.10.2001, 19:48:08
08.10.2001, 22:08:47	08.10.2001, 23:59:56	09.10.2001, 02:46:29
09.10.2001, 10:28:10	09.10.2001, 19:47:21	09.10.2001, 23:20:03
10.10.2001, 16:56:22	10.10.2001, 19:45:04	10.10.2001, 22:55:08
04.11.2003, 00:03:00		
`http://www.nur-ein-euro.de/`		
24.05.2002, 18:15:02	05.08.2002, 00:45:22	04.11.2003, 00:22:16

Fortsetzung auf nächster Seite

Fortsetzung von letzter Seite

`http://www.opel.de/`		
12.12.1998, 03:16:18	02.03.2000, 10:12:41	08.04.2000, 14:36:08
10.05.2000, 01:03:52	20.05.2000, 06:27:05	21.06.2000, 09:38:14
07.07.2000, 00:05:34	07.10.2000, 20:43:53	18.10.2000, 17:59:38
09.11.2000, 11:03:00	06.12.2000, 00:28:00	05.01.2001, 00:01:00
18.01.2001, 23:27:00	01.03.2001, 19:06:01	31.03.2001, 10:42:39
18.04.2001, 15:33:03	16.05.2001, 03:50:54	19.06.2001, 17:11:57
09.07.2001, 02:06:49	24.01.2002, 07:54:30	02.08.2002, 05:57:17
03.11.2003, 23:29:41		
`http://www.partypaket.de/`		
04.12.2000, 15:45:00	20.02.2003, 04:20:22	04.11.2003, 00:20:08
`http://www.prosieben.de/home/`		
02.08.2002, 22:16:19	14.10.2002, 15:11:32	12.11.2002, 21:58:42
08.12.2002, 21:40:14	01.02.2003, 08:53:51	03.11.2003, 23:25:59
`http://www.racheshop.com/`		
15.02.2003, 02:45:21	04.11.2003, 00:20:33	
`http://www.radio-essen.de/`		
21.07.2001, 18:14:43	23.09.2001, 00:48:41	26.11.2001, 20:22:22
22.01.2002, 19:43:45	29.05.2002, 16:13:53	04.06.2002, 13:17:10
02.08.2002, 07:58:31	03.08.2002, 01:20:52	28.09.2002, 08:15:44
29.09.2002, 23:50:19	25.11.2002, 00:06:00	19.02.2003, 00:42:32
03.11.2003, 23:35:30		
`http://www.sat1.de/`		
09.11.2000, 06:55:00	03.12.2000, 20:48:00	18.01.2001, 21:07:00
27.06.2001, 20:36:10	09.07.2001, 09:58:22	24.01.2002, 11:32:51
23.05.2002, 06:02:43	05.11.2002, 04:33:27	06.02.2003, 10:19:22
03.11.2003, 23:21:24		
`http://www.simpsonsweb.de/`		
23.07.2001, 07:07:58	02.08.2002, 13:54:20	28.09.2002, 11:05:46
29.11.2002, 11:48:59	01.02.2003, 17:53:44	18.02.2003, 18:39:57
04.11.2003, 00:33:00		
`http://www.strokes.de/`		
28.09.2002, 00:39:56	30.09.2002, 05:15:06	25.11.2002, 23:38:55
20.02.2003, 17:46:04	04.11.2003, 00:17:11	

Fortsetzung auf nächster Seite

Fortsetzung von letzter Seite

`http://www.virtuelle-apotheke.de/`		
16.08.2001, 05:06:19	01.10.2001, 13:21:40	24.05.2002, 00:52:12
27.05.2002, 21:48:44	03.06.2002, 22:12:14	02.08.2002, 21:06:53
06.08.2002, 15:55:34	28.09.2002, 16:53:10	30.09.2002, 10:48:00
27.01.2003, 21:59:02	15.02.2003, 05:34:54	04.11.2003, 00:19:41
`http://www.weltbild.de/`		
28.12.1996, 22:43:41	12.12.1998, 03:40:40	25.01.1999, 10:44:55
03.02.1999, 00:32:16	30.03.2001, 05:48:44	05.04.2001, 14:25:35
15.05.2001, 22:05:02	20.05.2001, 09:03:28	01.06.2001, 00:01:03
22.06.2001, 16:11:08	28.05.2002, 17:38:05	02.06.2002, 05:36:06
05.06.2002, 16:28:44	02.08.2002, 15:55:19	30.09.2002, 00:02:04
27.11.2002, 17:36:47	22.01.2003, 16:35:13	04.11.2003, 00:32:28

Tabelle A.4: Elemente TestSet-2

Anhang B

Auswertungstabellen

In den folgenden Tabellen finden sich die Ergebnisse der Evaluation von der Testdatenmenge. Für Tests wurde Kreuzvalidierung mit fünf Teilen verwendet, der Lauf ist in der Tabelle mit angegeben. Sind für einige oder alle Läufe mit einem bestimmten Kernmaß keine Werte vorhanden, so ist der Lernlauf nicht innerhalb von 300 Sekunden zu einem Ergebnis gekommen. Diese Beschränkung war nötig, um die Gesamtzeit des Tests handhabbar zu machen.
In den einzelnen Spalten sind die Werte für Precision, Recall, Accuracy und F-Measure angegeben. Eine genaue Beschreibung der Performanzmaße ist in Kapitel 5.1 zu finden. Alle Maße wurden einzeln getestet, als auch in ausgewählten Kombinationen nach Typ. Außerdem wurden die Test-/Lernmengen gewichtet und ungewichtet verwendet, d.h. die Lern-/Testmengen wurden im gewichteten Lauf so angepasst, daß die Anzahl der negativen Beispiele gleich der Anzahl der positiven Beispiele war.

B.1 Testdatensatz 1

B.1.1 Zusammenfassung nach Kernmaß (ungewichtet)

Maß	Läufe	Precision	Recall	Accuracy	F-Measure
02 - LevenstheinStructureMeasure	3	1.000	0.871	0.978	0.931
03 - LevenstheinTextMeasure	1	N/V	0.000	0.835	N/V
04 - SimpleRawTextMeasure	5	0.167	0.798	0.300	0.276
05 - URLMeasure	5	1.000	1.000	1.000	1.000
07 - TextLinkMeasure	4	0.974	0.843	0.970	0.904
08 - SimpleTextMeasure	5	0.181	0.661	0.443	0.284
09 - URLLinkBagMeasure	1	1.000	0.882	0.980	0.938
10 - URLLinkMeasure	5	1.000	0.488	0.914	0.656
11 - EuklidAngleTextDocumentvector-Measure	2	1.000	0.896	0.983	0.945
12 - EuklidDistanceTextDocumentvec-torMeasure	3	N/V	0.000	0.834	N/V
13 - HeadingMeasure	3	0.887	0.851	0.957	0.869
15 - TextLinkSupersetMeasure	1	0.906	0.879	0.965	0.892
17 - URLQueryOnlyMeasure	5	N/V	0.000	0.833	N/V
18 - WordBagMeasure	1	N/V	0.000	0.835	N/V
alle	5	1.000	1.000	1.000	1.000
nur_link	5	1.000	0.946	0.991	0.972
nur_struktur	5	1.000	0.851	0.975	0.920
nur_text	5	1.000	0.935	0.989	0.966
ohne_URL	5	1.000	0.946	0.991	0.972

Tabelle B.1: TestSet-1: ungewichteter Testlauf, Kernfunktion linear

Maß	Läufe	Precision	Recall	Accuracy	F-Measure
01 - LevenstheinRawTextMeasure	5	1.000	0.923	0.987	0.960
02 - LevenstheinStructureMeasure	5	1.000	0.839	0.973	0.913
03 - LevenstheinTextMeasure	5	1.000	0.940	0.990	0.969
04 - SimpleRawTextMeasure	5	1.000	0.232	0.872	0.377
05 - URLMeasure	5	1.000	1.000	1.000	1.000
06 - TextLinkBagMeasure	5	1.000	0.946	0.991	0.972
07 - TextLinkMeasure	5	0.966	0.839	0.968	0.898
08 - SimpleTextMeasure	5	1.000	0.304	0.884	0.466
09 - URLLinkBagMeasure	5	1.000	0.935	0.989	0.966
10 - URLLinkMeasure	5	0.993	0.869	0.977	0.927
11 - EuklidAngleTextDocumentvector-Measure	5	1.000	0.881	0.980	0.937
12 - EuklidDistanceTextDocumentvectorMeasure	5	0.945	0.512	0.913	0.664
13 - HeadingMeasure	5	0.892	0.887	0.963	0.890
14 - URLLinkSupersetMeasure	5	0.975	0.940	0.986	0.958
15 - TextLinkSupersetMeasure	5	0.974	0.899	0.979	0.935
16 - URLQueryMeasure	5	1.000	1.000	1.000	1.000
17 - URLQueryOnlyMeasure	5	N/V	0.000	0.833	N/V
18 - WordBagMeasure	5	1.000	0.952	0.992	0.976
alle	5	1.000	1.000	1.000	1.000
nur_link	5	1.000	0.952	0.992	0.976
nur_struktur	5	1.000	0.833	0.972	0.909
nur_text	5	1.000	0.911	0.985	0.953
ohne_URL	5	1.000	0.923	0.987	0.960

Tabelle B.2: TestSet-1: ungewichteter Testlauf, Kernfunktion polynomiell

Maß	Läufe	Precision	Recall	Accuracy	F-Measure
01 - LevenstheinRawTextMeasure	5	1.000	0.923	0.987	0.960
02 - LevenstheinStructureMeasure	5	1.000	0.839	0.973	0.913
03 - LevenstheinTextMeasure	5	1.000	0.940	0.990	0.969
04 - SimpleRawTextMeasure	5	1.000	0.232	0.872	0.377
05 - URLMeasure	5	1.000	1.000	1.000	1.000
06 - TextLinkBagMeasure	5	1.000	0.946	0.991	0.972
07 - TextLinkMeasure	5	0.967	0.869	0.973	0.915
08 - SimpleTextMeasure	5	1.000	0.304	0.884	0.466
09 - URLLinkBagMeasure	5	1.000	0.935	0.989	0.966
10 - URLLinkMeasure	5	0.987	0.887	0.979	0.934
11 - EuklidAngleTextDocumentvector-Measure	5	1.000	0.881	0.980	0.937
12 - EuklidDistanceTextDocumentvec-torMeasure	5	0.948	0.548	0.919	0.694
13 - HeadingMeasure	5	0.882	0.887	0.961	0.884
14 - URLLinkSupersetMeasure	5	0.975	0.935	0.985	0.954
15 - TextLinkSupersetMeasure	5	0.974	0.905	0.980	0.938
16 - URLQueryMeasure	5	1.000	1.000	1.000	1.000
17 - URLQueryOnlyMeasure	5	N/V	0.000	0.833	N/V
18 - WordBagMeasure	5	1.000	0.952	0.992	0.976
alle	5	1.000	1.000	1.000	1.000
nur_link	5	1.000	0.952	0.992	0.976
nur_struktur	5	1.000	0.875	0.979	0.933
nur_text	5	1.000	0.940	0.990	0.969
ohne_URL	5	1.000	0.946	0.991	0.972

Tabelle B.3: TestSet-1: ungewichteter Testlauf, Kernfunktion radial

Maß	Läufe	Precision	Recall	Accuracy	F-Measure
01 - LevenstheinRawTextMeasure	5	0.969	0.940	0.985	0.955
02 - LevenstheinStructureMeasure	5	0.793	0.887	0.942	0.837
03 - LevenstheinTextMeasure	2	1.000	0.939	0.990	0.969
04 - SimpleRawTextMeasure	5	1.000	0.232	0.872	0.377
06 - TextLinkBagMeasure	5	0.958	0.958	0.986	0.958
07 - TextLinkMeasure	5	0.910	0.899	0.968	0.904
08 - SimpleTextMeasure	5	1.000	0.304	0.884	0.466
09 - URLLinkBagMeasure	5	0.975	0.946	0.987	0.961
10 - URLLinkMeasure	5	0.980	0.887	0.978	0.931
11 - EuklidAngleTextDocumentvector-Measure	5	1.000	0.887	0.981	0.940
12 - EuklidDistanceTextDocumentvectorMeasure	5	0.938	0.089	0.847	0.163
13 - HeadingMeasure	2	N/V	0.000	0.833	N/V
14 - URLLinkSupersetMeasure	5	0.947	0.952	0.983	0.950
15 - TextLinkSupersetMeasure	5	0.909	0.952	0.976	0.930
17 - URLQueryOnlyMeasure	5	N/V	0.000	0.833	N/V
18 - WordBagMeasure	5	1.000	0.958	0.993	0.979
nur_struktur	5	0.759	0.750	0.918	0.754
nur_text	3	0.971	0.657	0.939	0.784

Tabelle B.4: TestSet-1: ungewichteter Testlauf, Kernfunktion sigmoid

B.1.2 Zusammenfassung nach Kernmaß (gewichtet)

Maß	Läufe	Precision	Recall	Accuracy	F-Measure
02 - LevenstheinStructureMeasure	1	1.000	0.824	0.912	0.903
04 - SimpleRawTextMeasure	5	0.500	1.000	0.500	0.667
06 - TextLinkBagMeasure	2	1.000	0.970	0.985	0.985
07 - TextLinkMeasure	5	0.532	0.685	0.542	0.599
08 - SimpleTextMeasure	5	0.500	1.000	0.500	0.667
09 - URLLinkBagMeasure	3	1.000	0.900	0.950	0.947
10 - URLLinkMeasure	5	1.000	0.500	0.750	0.667
13 - HeadingMeasure	5	1.000	0.881	0.940	0.937
14 - URLLinkSupersetMeasure	2	0.984	0.940	0.963	0.962
17 - URLQueryOnlyMeasure	5	N/V	0.000	0.500	N/V
alle	5	1.000	1.000	1.000	1.000
nur_link	5	1.000	0.952	0.976	0.976
nur_struktur	1	1.000	0.909	0.955	0.952
nur_text	5	1.000	0.929	0.964	0.963
ohne_URL	5	1.000	0.958	0.979	0.979

Tabelle B.5: TestSet-1: gewichteter Testlauf, Kernfunktion linear

Maß	Läufe	Precision	Recall	Accuracy	F-Measure
01 - LevenstheinRawTextMeasure	5	1.000	0.923	0.961	0.960
02 - LevenstheinStructureMeasure	5	1.000	0.845	0.923	0.916
03 - LevenstheinTextMeasure	5	1.000	0.935	0.967	0.966
04 - SimpleRawTextMeasure	5	1.000	0.232	0.616	0.377
05 - URLMeasure	5	1.000	1.000	1.000	1.000
06 - TextLinkBagMeasure	5	1.000	0.946	0.973	0.972
07 - TextLinkMeasure	5	0.993	0.804	0.899	0.888
08 - SimpleTextMeasure	5	1.000	0.304	0.652	0.466
09 - URLLinkBagMeasure	5	1.000	0.899	0.949	0.947
10 - URLLinkMeasure	5	1.000	0.792	0.896	0.884
11 - EuklidAngleTextDocumentvector-Measure	5	0.993	0.899	0.946	0.944
12 - EuklidDistanceTextDocumentvec-torMeasure	4	0.984	0.448	0.720	0.615
13 - HeadingMeasure	5	1.000	0.881	0.940	0.937
14 - URLLinkSupersetMeasure	5	0.994	0.946	0.970	0.970
15 - TextLinkSupersetMeasure	5	0.988	0.952	0.970	0.970
16 - URLQueryMeasure	5	1.000	1.000	1.000	1.000
17 - URLQueryOnlyMeasure	5	N/V	0.000	0.500	N/V
18 - WordBagMeasure	5	1.000	0.940	0.970	0.969
alle	5	1.000	1.000	1.000	1.000
nur_link	5	1.000	0.929	0.964	0.963
nur_struktur	5	1.000	0.893	0.946	0.943
nur_text	5	1.000	0.905	0.952	0.950
ohne_URL	5	1.000	0.905	0.952	0.950

Tabelle B.6: TestSet-1: gewichteter Testlauf, Kernfunktion polynomiell

Maß	Läufe	Precision	Recall	Accuracy	F-Measure
01 - LevenstheinRawTextMeasure	5	1.000	0.929	0.964	0.963
02 - LevenstheinStructureMeasure	5	1.000	0.857	0.929	0.923
03 - LevenstheinTextMeasure	5	1.000	0.940	0.970	0.969
04 - SimpleRawTextMeasure	5	1.000	0.232	0.616	0.377
05 - URLMeasure	5	1.000	1.000	1.000	1.000
06 - TextLinkBagMeasure	5	1.000	0.946	0.973	0.972
07 - TextLinkMeasure	5	0.986	0.863	0.926	0.921
08 - SimpleTextMeasure	5	1.000	0.304	0.652	0.466
09 - URLLinkBagMeasure	5	1.000	0.935	0.967	0.966
10 - URLLinkMeasure	5	1.000	0.845	0.923	0.916
11 - EuklidAngleTextDocumentvector-Measure	5	0.994	0.911	0.952	0.950
12 - EuklidDistanceTextDocumentvec-torMeasure	5	0.977	0.506	0.747	0.667
13 - HeadingMeasure	5	1.000	0.887	0.943	0.940
14 - URLLinkSupersetMeasure	5	0.994	0.952	0.973	0.973
15 - TextLinkSupersetMeasure	5	0.982	0.952	0.967	0.967
16 - URLQueryMeasure	5	1.000	1.000	1.000	1.000
17 - URLQueryOnlyMeasure	5	N/V	0.000	0.500	N/V
18 - WordBagMeasure	5	1.000	0.952	0.976	0.976
alle	5	1.000	1.000	1.000	1.000
nur_link	5	0.994	0.952	0.973	0.973
nur_struktur	5	1.000	0.917	0.958	0.957
nur_text	5	0.994	0.946	0.970	0.970
ohne_URL	5	0.988	0.964	0.976	0.976

Tabelle B.7: TestSet-1: gewichteter Testlauf, Kernfunktion radial

Maß	Läufe	Precision	Recall	Accuracy	F-Measure
01 - LevenstheinRawTextMeasure	5	1.000	0.929	0.964	0.963
02 - LevenstheinStructureMeasure	5	0.878	0.899	0.887	0.888
03 - LevenstheinTextMeasure	5	1.000	0.940	0.970	0.969
04 - SimpleRawTextMeasure	5	1.000	0.232	0.616	0.377
05 - URLMeasure	5	0.994	1.000	0.997	0.997
06 - TextLinkBagMeasure	5	1.000	0.952	0.976	0.976
07 - TextLinkMeasure	5	0.986	0.863	0.926	0.921
08 - SimpleTextMeasure	5	1.000	0.304	0.652	0.466
09 - URLLinkBagMeasure	5	1.000	0.935	0.967	0.966
10 - URLLinkMeasure	5	1.000	0.845	0.923	0.916
11 - EuklidAngleTextDocumentvector-Measure	5	0.994	0.923	0.958	0.957
12 - EuklidDistanceTextDocumentvectorMeasure	4	0.982	0.400	0.696	0.568
13 - HeadingMeasure	5	1.000	0.887	0.943	0.940
14 - URLLinkSupersetMeasure	5	0.994	0.952	0.973	0.973
15 - TextLinkSupersetMeasure	5	0.982	0.958	0.970	0.970
16 - URLQueryMeasure	5	0.988	1.000	0.994	0.994
17 - URLQueryOnlyMeasure	5	0.500	1.000	0.500	0.667
18 - WordBagMeasure	5	1.000	0.952	0.976	0.976
nur_link	4	0.992	0.970	0.981	0.981
nur_struktur	4	0.989	0.689	0.841	0.812
nur_text	2	N/V	0.000	0.500	N/V

Tabelle B.8: TestSet-1: gewichteter Testlauf, Kernfunktion sigmoid

B.2 Testdatensatz 2

B.2.1 Zusammenfassung nach Kernmaß (ungewichtet)

Maß	Läufe	Precision	Recall	Accuracy	F-Measure
04 - SimpleRawTextMeasure	5	N/V	0.000	0.716	N/V
06 - TextLinkBagMeasure	1	1.000	0.911	0.975	0.953
08 - SimpleTextMeasure	5	0.287	0.204	0.631	0.238
09 - URLLinkBagMeasure	5	1.000	0.695	0.913	0.820
10 - URLLinkMeasure	5	1.000	0.544	0.871	0.705
11 - EuklidAngleTextDocumentvector-Measure	2	1.000	0.422	0.836	0.594
12 - EuklidDistanceTextDocumentvec-torMeasure	3	0.319	0.316	0.613	0.317
13 - HeadingMeasure	5	N/V	0.000	0.716	N/V
14 - URLLinkSupersetMeasure	5	0.627	0.850	0.814	0.722
15 - TextLinkSupersetMeasure	1	N/V	0.000	0.713	N/V
16 - URLQueryMeasure	5	1.000	1.000	1.000	1.000
17 - URLQueryOnlyMeasure	5	N/V	0.000	0.716	N/V
18 - WordBagMeasure	1	N/V	0.000	0.717	N/V
alle	5	1.000	0.942	0.984	0.970
nur_link	5	1.000	0.929	0.980	0.963
nur_struktur	5	0.990	0.876	0.962	0.930
nur_text	5	1.000	0.916	0.976	0.956
ohne_URL	5	1.000	0.934	0.981	0.966

Tabelle B.9: TestSet-2: ungewichteter Testlauf, Kernfunktion linear

Maß	Läufe	Precision	Recall	Accuracy	F-Measure
01 - LevenstheinRawTextMeasure	5	1.000	0.881	0.966	0.936
02 - LevenstheinStructureMeasure	5	0.990	0.872	0.961	0.927
03 - LevenstheinTextMeasure	5	1.000	0.912	0.975	0.954
04 - SimpleRawTextMeasure	5	1.000	0.018	0.721	0.035
05 - URLMeasure	5	1.000	1.000	1.000	1.000
06 - TextLinkBagMeasure	5	1.000	0.912	0.975	0.954
07 - TextLinkMeasure	5	0.973	0.805	0.938	0.881
08 - SimpleTextMeasure	5	1.000	0.190	0.770	0.320
09 - URLLinkBagMeasure	5	1.000	0.867	0.962	0.929
10 - URLLinkMeasure	5	1.000	0.788	0.940	0.881
11 - EuklidAngleTextDocumentvector-Measure	5	0.985	0.894	0.966	0.937
13 - HeadingMeasure	5	N/V	0.000	0.716	N/V
14 - URLLinkSupersetMeasure	5	1.000	0.929	0.980	0.963
15 - TextLinkSupersetMeasure	5	1.000	0.934	0.981	0.966
16 - URLQueryMeasure	5	1.000	1.000	1.000	1.000
17 - URLQueryOnlyMeasure	5	N/V	0.000	0.716	N/V
18 - WordBagMeasure	5	1.000	0.925	0.979	0.961
alle	5	1.000	0.951	0.986	0.975
nur_link	5	1.000	0.925	0.979	0.961
nur_struktur	5	0.990	0.867	0.960	0.925
nur_text	5	1.000	0.903	0.972	0.949
ohne_URL	5	1.000	0.920	0.977	0.959

Tabelle B.10: TestSet-2: ungewichteter Testlauf, Kernfunktion polynomiell

Maß	Läufe	Precision	Recall	Accuracy	F-Measure
01 - LevenstheinRawTextMeasure	5	1.000	0.881	0.966	0.936
02 - LevenstheinStructureMeasure	5	0.990	0.872	0.961	0.927
03 - LevenstheinTextMeasure	5	1.000	0.920	0.977	0.959
04 - SimpleRawTextMeasure	5	1.000	0.018	0.721	0.035
05 - URLMeasure	5	1.000	1.000	1.000	1.000
06 - TextLinkBagMeasure	5	0.995	0.912	0.974	0.952
07 - TextLinkMeasure	5	0.963	0.810	0.937	0.880
08 - SimpleTextMeasure	5	1.000	0.190	0.770	0.320
09 - URLLinkBagMeasure	5	1.000	0.898	0.971	0.946
10 - URLLinkMeasure	5	1.000	0.801	0.943	0.889
11 - EuklidAngleTextDocumentvector-Measure	5	0.990	0.885	0.965	0.935
12 - EuklidDistanceTextDocumentvec-torMeasure	5	1.000	0.265	0.791	0.420
13 - HeadingMeasure	5	N/V	0.000	0.716	N/V
14 - URLLinkSupersetMeasure	5	1.000	0.934	0.981	0.966
15 - TextLinkSupersetMeasure	5	1.000	0.934	0.981	0.966
16 - URLQueryMeasure	5	1.000	1.000	1.000	1.000
17 - URLQueryOnlyMeasure	5	N/V	0.000	0.716	N/V
18 - WordBagMeasure	5	1.000	0.929	0.980	0.963
alle	5	1.000	0.991	0.997	0.996
nur_link	5	1.000	0.951	0.986	0.975
nur_struktur	5	0.990	0.867	0.960	0.925
nur_text	5	1.000	0.925	0.979	0.961
ohne_URL	5	1.000	0.938	0.982	0.968

Tabelle B.11: TestSet-2: ungewichteter Testlauf, Kernfunktion radial

Maß	Läufe	Precision	Recall	Accuracy	F-Measure
01 - LevenstheinRawTextMeasure	5	1.000	0.894	0.970	0.944
02 - LevenstheinStructureMeasure	4	0.819	0.674	0.865	0.739
03 - LevenstheinTextMeasure	5	1.000	0.920	0.977	0.959
04 - SimpleRawTextMeasure	5	1.000	0.018	0.721	0.035
06 - TextLinkBagMeasure	5	0.986	0.929	0.976	0.957
07 - TextLinkMeasure	3	0.910	0.816	0.925	0.860
08 - SimpleTextMeasure	5	1.000	0.190	0.770	0.320
09 - URLLinkBagMeasure	5	1.000	0.907	0.974	0.951
10 - URLLinkMeasure	5	1.000	0.805	0.945	0.892
11 - EuklidAngleTextDocumentvector-Measure	4	0.988	0.906	0.970	0.945
12 - EuklidDistanceTextDocumentvectorMeasure	3	1.000	0.265	0.791	0.419
13 - HeadingMeasure	5	N/V	0.000	0.716	N/V
14 - URLLinkSupersetMeasure	5	1.000	0.934	0.981	0.966
15 - TextLinkSupersetMeasure	5	1.000	0.947	0.985	0.973
17 - URLQueryOnlyMeasure	5	N/V	0.000	0.716	N/V
18 - WordBagMeasure	5	1.000	0.934	0.981	0.966
nur_link	3	0.936	0.978	0.975	0.957
nur_text	2	1.000	0.900	0.972	0.947
ohne_URL	1	1.000	0.933	0.981	0.966

Tabelle B.12: TestSet-2: ungewichteter Testlauf, Kernfunktion sigmoid

B.2.2 Zusammenfassung nach Kernmaß (gewichtet)

Maß	Läufe	Precision	Recall	Accuracy	F-Measure
04 - SimpleRawTextMeasure	5	N/V	0.000	0.500	N/V
06 - TextLinkBagMeasure	1	N/V	0.000	0.500	N/V
08 - SimpleTextMeasure	5	N/V	0.000	0.500	N/V
09 - URLLinkBagMeasure	5	0.995	0.925	0.960	0.959
10 - URLLinkMeasure	5	1.000	0.845	0.923	0.916
11 - EuklidAngleTextDocumentvector-Measure	2	1.000	0.456	0.728	0.626
12 - EuklidDistanceTextDocumentvec-torMeasure	1	N/V	0.000	0.500	N/V
13 - HeadingMeasure	5	0.560	0.973	0.604	0.711
14 - URLLinkSupersetMeasure	5	0.995	0.912	0.954	0.952
16 - URLQueryMeasure	5	1.000	1.000	1.000	1.000
17 - URLQueryOnlyMeasure	5	N/V	0.000	0.500	N/V
alle	5	1.000	0.938	0.969	0.968
nur_link	5	1.000	0.929	0.965	0.963
nur_struktur	5	0.995	0.881	0.938	0.934
nur_text	5	1.000	0.925	0.962	0.961
ohne_URL	5	1.000	0.929	0.965	0.963

Tabelle B.13: TestSet-2: gewichteter Testlauf, Kernfunktion linear

Maß	Läufe	Precision	Recall	Accuracy	F-Measure
01 - LevenstheinRawTextMeasure	5	1.000	0.881	0.940	0.936
02 - LevenstheinStructureMeasure	5	0.995	0.872	0.934	0.929
03 - LevenstheinTextMeasure	5	1.000	0.912	0.956	0.954
04 - SimpleRawTextMeasure	5	1.000	0.018	0.509	0.035
05 - URLMeasure	5	1.000	1.000	1.000	1.000
06 - TextLinkBagMeasure	5	1.000	0.912	0.956	0.954
07 - TextLinkMeasure	5	0.995	0.823	0.909	0.901
08 - SimpleTextMeasure	5	1.000	0.190	0.595	0.320
09 - URLLinkBagMeasure	5	1.000	0.832	0.916	0.908
10 - URLLinkMeasure	5	1.000	0.783	0.892	0.878
11 - EuklidAngleTextDocumentvector-Measure	5	0.986	0.925	0.956	0.954
12 - EuklidDistanceTextDocumentvec-torMeasure	1	N/V	0.000	0.500	N/V
13 - HeadingMeasure	5	0.560	0.973	0.604	0.711
14 - URLLinkSupersetMeasure	5	1.000	0.894	0.947	0.944
15 - TextLinkSupersetMeasure	5	1.000	0.934	0.967	0.966
16 - URLQueryMeasure	5	1.000	1.000	1.000	1.000
17 - URLQueryOnlyMeasure	5	N/V	0.000	0.500	N/V
18 - WordBagMeasure	5	1.000	0.925	0.962	0.961
alle	5	1.000	0.934	0.967	0.966
nur_link	5	1.000	0.916	0.958	0.956
nur_struktur	5	0.995	0.872	0.934	0.929
nur_text	5	1.000	0.916	0.958	0.956
ohne_URL	5	1.000	0.920	0.960	0.959

Tabelle B.14: TestSet-2: gewichteter Testlauf, Kernfunktion polynomiell

Maß	Läufe	Precision	Recall	Accuracy	F-Measure
01 - LevenstheinRawTextMeasure	5	1.000	0.881	0.940	0.936
02 - LevenstheinStructureMeasure	5	0.995	0.872	0.934	0.929
03 - LevenstheinTextMeasure	4	1.000	0.934	0.967	0.966
04 - SimpleRawTextMeasure	5	1.000	0.018	0.509	0.035
05 - URLMeasure	5	1.000	1.000	1.000	1.000
06 - TextLinkBagMeasure	5	1.000	0.912	0.956	0.954
07 - TextLinkMeasure	5	0.989	0.823	0.907	0.899
08 - SimpleTextMeasure	5	1.000	0.190	0.595	0.320
09 - URLLinkBagMeasure	5	1.000	0.858	0.929	0.924
10 - URLLinkMeasure	5	1.000	0.788	0.894	0.881
11 - EuklidAngleTextDocumentvector-Measure	5	0.995	0.903	0.949	0.947
12 - EuklidDistanceTextDocumentvec-torMeasure	4	1.000	0.238	0.619	0.384
13 - HeadingMeasure	5	0.560	0.973	0.604	0.711
14 - URLLinkSupersetMeasure	5	1.000	0.925	0.962	0.961
15 - TextLinkSupersetMeasure	5	1.000	0.934	0.967	0.966
16 - URLQueryMeasure	5	1.000	1.000	1.000	1.000
17 - URLQueryOnlyMeasure	5	N/V	0.000	0.500	N/V
18 - WordBagMeasure	5	1.000	0.929	0.965	0.963
alle	5	1.000	0.996	0.998	0.998
nur_link	5	1.000	0.938	0.969	0.968
nur_struktur	5	0.995	0.872	0.934	0.929
nur_text	5	1.000	0.925	0.962	0.961
ohne_URL	5	1.000	0.938	0.969	0.968

Tabelle B.15: TestSet-2: gewichteter Testlauf, Kernfunktion radial

Maß	Läufe	Precision	Recall	Accuracy	F-Measure
01 - LevenstheinRawTextMeasure	5	1.000	0.881	0.940	0.936
02 - LevenstheinStructureMeasure	5	0.870	0.916	0.889	0.892
03 - LevenstheinTextMeasure	5	1.000	0.920	0.960	0.959
04 - SimpleRawTextMeasure	5	1.000	0.018	0.509	0.035
06 - TextLinkBagMeasure	5	1.000	0.916	0.958	0.956
07 - TextLinkMeasure	5	0.989	0.823	0.907	0.899
08 - SimpleTextMeasure	5	1.000	0.190	0.595	0.320
09 - URLLinkBagMeasure	5	1.000	0.867	0.934	0.929
10 - URLLinkMeasure	5	1.000	0.788	0.894	0.881
11 - EuklidAngleTextDocumentvector-Measure	3	N/V	0.000	0.500	N/V
12 - EuklidDistanceTextDocumentvectorMeasure	5	1.000	0.155	0.577	0.268
13 - HeadingMeasure	5	0.560	0.973	0.604	0.711
14 - URLLinkSupersetMeasure	5	1.000	0.929	0.965	0.963
15 - TextLinkSupersetMeasure	5	0.968	0.947	0.958	0.957
17 - URLQueryOnlyMeasure	5	N/V	0.000	0.500	N/V
18 - WordBagMeasure	5	1.000	0.929	0.965	0.963
nur_link	5	0.986	0.960	0.973	0.973
nur_struktur	1	N/V	0.000	0.500	N/V
ohne_URL	1	1.000	0.891	0.946	0.943

Tabelle B.16: TestSet-2: gewichteter Testlauf, Kernfunktion sigmoid