Martin-Luther-Universität Halle-Wittenberg
Naturwissenschaftliche Fakultät III
Institut für Informatik

Prof. Dr. Stefan Braß, Dr. Henning Thielemann

10. Übung zur Vorlesung „Grundlagen des WWW“

Sommersemester 2009 — Ausgabe: 2009-06-18

Abgabe: 2009-06-25

Aufgabe 10.1: Datenextraktion (10 Punkte)

Am 2009-09-27 ist Bundestagwahl und mithin wieder die große Frage, wem man seine Stimme gibt. Möglicherweise kennen Sie bereits den http://www.wahl-o-mat.de/ von der Bundeszentrale für politische Bildung. Der Wahlomat vergleicht Parteien anhand von Themen, bei denen sich die Haltung der Parteien deutlich unterscheidet. Die Meinung des Wählers zu diesen Themen wird mit den Vorhaben in den Wahlprogrammen verglichen und daraus eine Übersicht erstellt, die zeigt, mit welcher Partei ein Wähler am meisten übereinstimmt.
Die Verwendung des Wahlprogrammes als Grundlage für den Vergleich hat den Vorteil, dass alle antretenden Parteien erfasst werden. Ein gravierender Nachteil ist allerdings, dass auf diese Weise die tatsächliche Politik unberücksichtigt bleibt. Erfahrungsgemäß weicht die tatsächliche Politik mitunter deutlich von den Wahlprogrammen ab oder Wahlprogramme werden in einer unerwarteten Weise umgesetzt. Auch ist der einzelne Politiker nicht an das Wahlprogramm gebunden. Der Volksmund hat für Politiker, die für eine Sache werben, aber gegen sie stimmen (oder umgekehrt) das schöne Wort „Umfaller“ geprägt.
Wie kann man den Wähler mit einem Programm unterstützen, welches diese Probleme berücksichtigt? Man könnte zum Beispiel die Meinung des Wählers dem tatsächlichen Abstimmverhalten der Bundestagsabgeordneten gegenüberstellen. Das Abstimmverhalten einzelner Bundestagsabgeordneter erfahren wir allerdings nur bei einer namentlichen Abstimmung. Dies muss laut §52 der Geschäftsordnung des Bundestages von einer Fraktion oder von mindestens 5 Prozent der anwesenden Abgeordneten beantragt werden. Seit Mai 2006 werden die namentlichen Abstimmungen vom Parlamentwatch e.V. im WWW veröffentlicht:

http://www.abgeordnetenwatch.de/abstimmungen-346-0.html .

Die Idee ist, dem Wähler die Liste der namentlichen Abstimmungen der vergangenen Legislaturperiode vorzulegen, ihn abstimmen zu lassen und dies mit den Abstimmungen der Abgeordneten zu vergleichen. Auf diese Weise kann ein Wähler diejenigen Kandidaten finden, die seine Meinung am ehesten vertreten, und in welchen Parteien diese Kandidaten sind. Dieses Verfahren ist, wie gesagt, auf Parteien und Kandidaten beschränkt, die bereits im Bundestag vertreten sind.
In einem ersten Schritt sollen Sie die Abstimmungsergebnisse auswerten. Laden Sie sich dazu von einer Abstimmung, die sie interessiert, die Seiten mit den Abstimmungsergebnissen herunter. Beispiel: Für die Abstimmung http://www.abgeordnetenwatch.de/banken_rettungspaket-636-156.html finden Sie alle Zustimmungen unter http://www.abgeordnetenwatch.de/banken_rettungspaket-636-156----abst_ja.html . Schreiben Sie nun ein Programm, welches aus den HTML-Abstimmungsdokumenten die Stimmen (ja, nein, enthalten, nicht beteiligt) der Abgeordneten herausfiltert. Die Ausgabe sollen SQL-Anweisungen zum Einfügen von Tupeln in eine Datenbank sein, welche zu folgender Tabellendefinition passen:

```
CREATE TABLE Voting (
   topic VARCHAR(60) NOT NULL,
   politician VARCHAR(60) NOT NULL,
   vote CHAR(4) NOT NULL CHECK (vote IN ('ja', 'nein', 'ent', 'nb')),
   PRIMARY KEY (topic, politician)
);
```

Wir wollen den Stamm des Dateinamens des HTML-Dokumentes eines Politikers auf `abgeordnetenwatch.de` als Schlüssel verwenden. Die Zuordnung dieses Schlüssels zu Name und Partei heben wir uns für später auf.

Aufruf und Ausgabe des Programmes sollten wie folgt aussehen:

```
$ wget http://www.abgeordnetenwatch.de/banken_rettungspaket-636-156----abst_ja.html
...
$ analyse-voting banken_rettungspaket-636-156
INSERT INTO Voting VALUES
   ('banken_rettungspaket-636-156', 'achim_grossmann-650-5769',  'ja');
INSERT INTO Voting VALUES
   ('banken_rettungspaket-636-156', 'albert_rupprecht-650-5526', 'ja');
...
INSERT INTO Voting VALUES
   ('banken_rettungspaket-636-156', 'alexander_bonde-650-5831',  'nein');
...
```

Natürlich ist es insgesamt gesehen ineffizient, dass `abgeordnetenwatch.de` aus einer kompakten Datenhaltung in einer Datenbank große HTML-Dokumente erzeugt, die über das Netz übertragen und dann von uns wieder auf eine kompakte Form reduziert werden. Wir wollen hier aber zu Übungszwecken davon ausgehen, dass die Daten nur in Form von HTML zu beschaffen sind.
Zum Filtern der HTML-Dokumente können Sie sich verschiedener Methoden bedienen. Die folgende Liste soll nur Anregungen geben. Wie und mit welcher Programmiersprache Sie die Aufgabe lösen, bleibt Ihre Entscheidung.

- Bibliotheken zur Verarbeitung von „Tag-Suppen“
  - Java: http://www.tagsoup.info/
  - Haskell: http://community.haskell.org/~ndm/tagsoup/, http://hackage.haskell.org/package/tagchup/
  - Python: `sgmllib/SGMLParser`

  Leider ist HTML-Code meistens nicht syntaktisch korrekt, so auch der Code von `abgeordnetenwatch.de`. Daher müssen Bibliotheken zur Verarbeitung von HTML diesem Umstand Rechnung tragen. Man versucht dabei gar nicht, HTML-Code zu korrigieren, da jede Interpretation von fehlerhaftem HTML-Code von der Absicht des Autors abweichen kann. Stattdessen interpretiert man HTML-Dokumente als eine Folge von HTML-Tags und Text, eben einer „TagSoup“.

  Es hat sich gezeigt, dass sich in dieser Darstellung automatisch generierte Seiten gut wieder auseinandernehmen lassen. Man formuliert damit Algorithmen wie „gib mir den Text, der nach dem `td`-Element kommt, welches das Attribut `'class="abgeordneter"'` enthält“.

- Bibliotheken zur Verarbeitung von XML/HTML-Bäumen mit einem fehlerkorrigierenden HTML-Parser (beispielsweise `hxt` in Haskell)

- Reguläre Ausdrücke in Perl oder Python