

**Proseminar Künstliche Intelligenz
SS2004
Intelligentes Datensammeln im Web: OntoMiner**

Stefan Kruppa
sk18@informatik.uni-ulm.de

Zusammenfassung

OntoMiner ist ein Programm, welches mit Hilfe verschiedener Algorithmen Webseiten nach Schlüsselbegriffen durchsucht und diese in eine hierarchische Gliederung überführt. Dies wird dadurch erreicht, dass dem Miner eine Menge von Webseiten übergeben wird, welche dann zunächst mittels Syntaxanalyse in halbstrukturierte XML-Dokumente überführt werden. Anschließend erfolgt mittels diverser Verfahren ein Ableiten von hierarchisch strukturierten Taxonomie-Bäumen. Schlüsselbegriffe und ihre Beziehungen können identifiziert werden.

Inhalt

1. Begriff, Zielsetzung und Entwickler	2
2. Semantisches Partitionieren	3
2.1 Flat Partitioner	3
2.2 Ermitteln der Datentabellen	5
2.3 Hierarchisches Partitionieren	5
3. Taxonomy Mining	9
3.1 Häufigkeitsbasiertes Mining	10
3.2 Extraktion weiterer in Frage kommender Labels	10
3.3 Kürzungsphase	10
3.4 Gruppieren der Labels in Konzepte	10
3.5 Herausarbeiten von „is-a“-Beziehungen aus hierarchisch partitionierten Webseiten	10
3.6 Erweitern der Taxonomie auf gesamte Websites	10
4. Instanzenextraktion	11
4.1 Identifizieren des Instanzsegments	11
4.2 Extrahieren von Attributen und Werten aus Instanzsegmenten	11
5. Experimentell erlangte Ergebnisse	11
6. Bewertung	12
7. Ausblick: Was ist heute möglich ?	12
8. Literatur	12

1. Begriff, Zielsetzung und Entwickler

Der Begriff OntoMiner ist eine Abkürzung für Ontology Miner. Ontologie wird in der Wikipedia Online-Enzyklopädie (WOE) [2] folgendermaßen erklärt:

„Unter einer Ontologie versteht man in der Informatik im Bereich Künstliche Intelligenz ein formal definiertes System von Dingen und/oder Konzepten und Relationen zwischen diesen Dingen. Zusätzlich enthalten Ontologien (zumindest implizit) Regeln“

und

„Eine Ontologie lässt sich vergleichen mit einer Datenbank – Struktur (Datenbankschema) und Inhalt (Daten) bilden ein Ganzes“

Vereinfacht gesagt beschäftigt sich OntoMiner also damit, Begriffe zu ordnen und Beziehungen zwischen ihnen „auszugraben“, also herauszuarbeiten. Zu diesem Zweck wird dem Programm zuerst eine Menge von etwa 10-15 Webseiten übergeben, die zusammen ein bestimmtes Fachgebiet beschreiben. Diese werden dann in einen XML-Baum (XML = Extended Markup Language, eine populäre Struktursprache, auf der unter anderem auch HTML basiert) umgeformt und anschließend einer Reihe von Verfahren unterzogen, die zum Schluss einen Baum ausgeben sollen, der die Schlüsselbegriffe der Seiten hierarchisch ordnet. Dieser Baum stellt dann eine Taxonomie dar. Die WOE [2] liefert wieder eine Definition für „Taxonomie“:

„In der Linguistik beschäftigt sich die Taxonomie mit der Segmentierung und Klassifikation sprachlicher Einheiten, um mit diesen ein Sprachsystem zu beschreiben. Auch andere Fachbereiche verwenden den Begriff der Taxonomie allgemein für ein Klassifikationssystem, eine Systematik oder den Vorgang des Klassifizierens.“

Zu beachten ist, dass das Programm nicht fähig ist, die Semantik der Begriffe zu erfassen; es bezieht seine Informationen, die es zur Hierarchisierung benötigt, stattdessen allein aus der Syntax der Quelltexte der HTML-Dokumente. Dies ist deshalb möglich, weil Inhalte, die für den Menschen als zusammengehörig auf einer Webseite kenntlich gemacht werden sollen, eine ähnliche Form besitzen (z.B. werden die einzelnen Links einer Navigationsleiste in der selben Tabelle stehen und mit den selben (oder zumindest sehr ähnlichen) Attributen versehen sein).

Das Ziel, das mit dem Programm erreicht werden soll, ist es, Grundlagen für das Semantic Web zu schaffen. Das Semantic Web soll ermöglichen, Suchmaschinen zu entwickeln, die bei der Eingabe eines Begriffs nicht nur lexikographische Übereinstimmungen finden, sondern den Kontext zu einem Term anzeigen. Beispielsweise soll eine Suche nach „Kohlrabi“ nicht nur die Seiten als Treffer liefern, in denen genau dieses Wort vorkommt, sondern auch Seiten über Gemüse oder Artikel über gesunde Ernährung. Da aber der überwiegende Anteil der im Internet verfügbaren Informationen nicht in strukturierter Form mitsamt Angaben über den Kontext vorliegen und eine manuelle Strukturierung aufgrund der riesigen und sich ständig ändernden Datenmengen unmöglich ist, muss zum Erreichen dieses Ziels ein Weg gefunden werden, auf automatisierte Weise diese Informationen so zu ordnen, dass sie vor allem maschinell weiterverarbeitet werden können.

Entwickelt wurde und wird OntoMiner vom Forschungsteam Hasan Davulcu, Srinivas Vadrevu, Saravanakumar Nagarajan an der Arizona State University in den USA. Ihr Paper, das ihre Arbeit dokumentiert, trägt den Titel „OntoMiner: Bootstrapping Ontologies From Overlapping Domain Specific Web sites“ [1] und dient dieser Arbeit als Grundlage.

In OntoMiner sind folgende Verfahren implementiert:

1. Semantisches Partitionieren (wandelt Webseiten in semantische Bäume um)
2. Taxonomy Mining (untersucht die semantischen Bäume nach häufigen Labels, gruppiert diese in Konzepte und strukturiert diese Konzepte)
3. Instanzenextraktion („füllt“ die Konzepte mit passenden Begriffen (Instanzen))

Jedes dieser Verfahren, die im folgenden erläutert werden, besteht aus einer Folge von Einzelschritten.

2. Semantisches Partitionieren

Zuerst werden dem Programm etwa 10-15 URLs von Webseiten übergeben, die verarbeitet werden sollen. Diese Seiten müssen drei Voraussetzungen erfüllen:

Sie müssen:

- taxonomie-orientiert sein, also einer bestimmten Systematik folgen
- sich „überlappen“, also müssen zumindest einige der Schlüsselbegriffe mehrfach auftreten
- das Interessengebiet charakterisieren

Diese Seiten werden nun vom Semantischen Partitionierer in mehreren Schritten aufbereitet. Zu Beginn wird jede Seite in ihre logischen Segmente aufgeteilt; danach werden die Tabellen identifiziert, die tatsächliche Nutzdaten enthalten (und nicht nur designtechnisches Hilfsmittel sind) und aus diesen jene Nutzdaten extrahiert. Falls in einer Seite keine solchen Datentabellen enthalten sind, wird der aufwendigere Hierarchische Partitionierer herangezogen, der in mehreren Phasen die Inhalte des XML-Baumes ausliest und strukturiert.

2.1 Flat Partitioner

Im ersten Schritt unterteilt der Flat Partitioner eine Webseite in ihre logischen Segmente auf, wie zum Beispiel Kopfzeile, Navigationsleiste und Textfeld. Wie dies erreicht wird werde ich am folgenden Beispiel, der Webseite der New York Times, darstellen:



Abbildung 1: Aufteilung einer Webseite in Segmente (Beispiel)

Die Seite wird in logische Segmente (im folgenden auch Blöcke genannt) unterteilt, die in Abbildung 1 umrandet und mit B1 bis B5 beschriftet sind. Zu diesem Zweck betrachtet man den XML-Syntaxbaum, der aus dem HTML-Quelltext erzeugt wird (Abbildung 2).

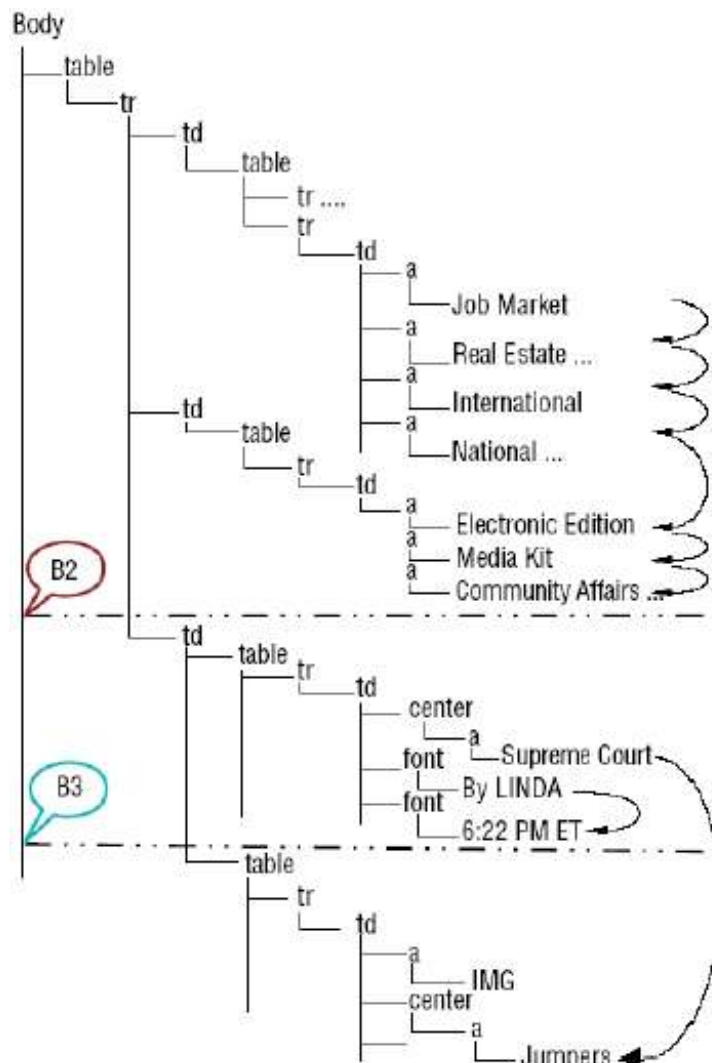


Abbildung 2: XML-Syntaxbaum der Beispiel-Webseite

Bis zur ersten gestrichelten Linie stehen Struktur und Inhalt der Navigationsleiste (Blöcke B1 und B2), zwischen erster und zweiter Linie die des Textfelds (Block B3). Man sieht, dass hier vor allem verschachtelte Tabellen zur Strukturierung verwendet werden, in denen dann die Links zu den einzelnen Bereichen der Webseite stehen. Die einzelnen Einträge wie z.B. „body“, „table“ und „td“ nennen wir, wie in Bäumen üblich, Knoten.

Ziel ist es nun, die Grenzen der einzelnen Blöcke zu finden.

Die Pfade zu „Job Market“, „Real Estate“, „International“ und „National“ sind jeweils *ähnlich*, weshalb sie mit einem „similarity link“ verbunden werden. Ähnlich heißt, dass man von der Wurzel aus die Pfade zu den Knoten entlangwandern kann und dabei Knoten der selben Tiefe die gleichen Namen tragen. Zusätzliche Bedingung für einen Link ist, dass Knoten, die zwischen den zu verbindenden Knoten liegen, unterschiedliche Pfade besitzen (zu sehen bei „Supreme Court“ und „Jumpers“; zwischen diesen beiden Blattknoten liegen nur Knoten, die nicht die Bezeichnung „center“ tragen).

Nun wird der erste Blattknoten in den Block aufgenommen und zugleich als vorläufige Blockgrenze festgelegt. Dann wird der Quotient von diesem Knoten ausgehender einzigartiger „similarity links“ zur Gesamtzahl der einzigartigen „similarity links“ im aktuellen Segment berechnet und mit einem vorher festgelegten Grenzwert verglichen. Ist der Quotient kleiner als der Grenzwert, so ist der betrachtete Knoten die neue Blockgrenze und der nächste Knoten wird zum Beginn des nächsten Blocks, ansonsten wird der Folgeknoten als mögliche Grenze untersucht.

Bsp.: Der Grenzwert sei 60%. „Job Market“ ist der erste untersuchte Knoten. Von ihm geht genau ein einzigartiger Link aus und die Gesamtzahl der einzigartigen Links im Segment ist ebenfalls eins. Der Quotient ist demnach 100%, also größer als der Grenzwert. Somit wird „Real Estate“ zum nächsten Kandidaten für die Blockgrenze. Ein „similarity link“ läuft aus dem Segment in seiner jetzigen Größe heraus (und zwar zu „International“), dieser ist momentan einzigartig. Da der Link von „Job Market“ auf „Real Estate“ zugleich ein Link in umgekehrter Richtung ist, ist dieser nicht mehr einzigartig. Somit bleibt nur ein einzigartiger Link übrig (der zu „International“). Damit ist der Quotient wieder 100%.

Bei „Community Affairs“ gibt es keinen herauslaufenden Link und genau einen einzigartigen. Somit wird der Quotient 0%, „Community Affairs“ wird zur Blockgrenze.

2.2 Ermitteln der Datentabellen

Wie bereits erwähnt, werden Tabellen oft zur Strukturierung und als Designmittel verwendet anstatt im ursprünglichen Sinne Daten zu veranschaulichen. Deshalb muss ein besonderer Aufwand betrieben werden, um die tatsächlichen Nutzdaten zu finden. In diesem Schritt werden nun die „echten“ Datentabellen bestimmt. Als Datentabellen werden solche Tabellen deklariert, die die folgenden beiden Bedingungen erfüllen:

- Der Wert des „border“-Attributs ist größer als Null (d.h. es handelt sich um eine Tabelle, die vom Browser umrandet wird und damit vom Betrachter auch als Tabelle erkannt werden kann, und nicht um eine „blinde“, also unsichtbare, Tabelle, die oftmals eingesetzt wird, um Absätze einheitlich auszurichten) oder die Tabelle enthält mindestens 2 Zeilen und 2 Spalten
- Paarweise aufeinander folgende Zeilen sind in Bezug auf den Gruppierungsaufwand ähnlich (s. Abschnitt 2.3)

Nachdem die Datentabellen gefunden wurden, wird nun die Orientierung jeder Tabelle bestimmt. Wenn die Werte der Kopfzeile häufig vorkommen, wird die Tabelle als spaltenorientiert eingestuft, wenn die Werte der Spalten häufig vorkommen als zeilenorientiert. Falls beides eintritt wird die Tabelle als Zeilen-Spalten-orientiert bezeichnet. Schließlich werden die Daten-Tupel entsprechend der Ausrichtung der Tabelle extrahiert.

Falls der Block keine Datentabellen enthält wird er vom, im folgenden erläuterten, hierarchischen Partitionierer behandelt.

2.3 Hierarchisches Partitionieren

Falls Datentabellen in einem Block vorhanden sind kann man daraus den Inhalt extrahieren. Falls dies aber nicht der Fall ist, so muss man auf andere Weise die Nutzinformationen auslesen. Dies geschieht durch Hierarchisches Partitionieren. Hierbei werden hierarchische Beziehungen zwischen den einzelnen Blattknoten (in denen ja der Inhalt der Webseite steht) abgeleitet.

Veranschaulicht sieht das Ziel in diesem Schritt wie in Abbildung 3 aus.

Wir haben den XML-Syntaxbaum wie in der linken Grafik (Abbildung 3a) dargestellt und wollen nun eine Gliederung erreichen wie sie in der rechten Grafik (Abbildung 3b) zu sehen ist. Zum Beispiel sind „News“ und „Opinion“ die Kategorien, denen die einzelnen Einträge untergeordnet sind, wir haben also eine Hierarchie erzeugt.



Abbildung 3a: Ausgangspunkt: XML-Baum

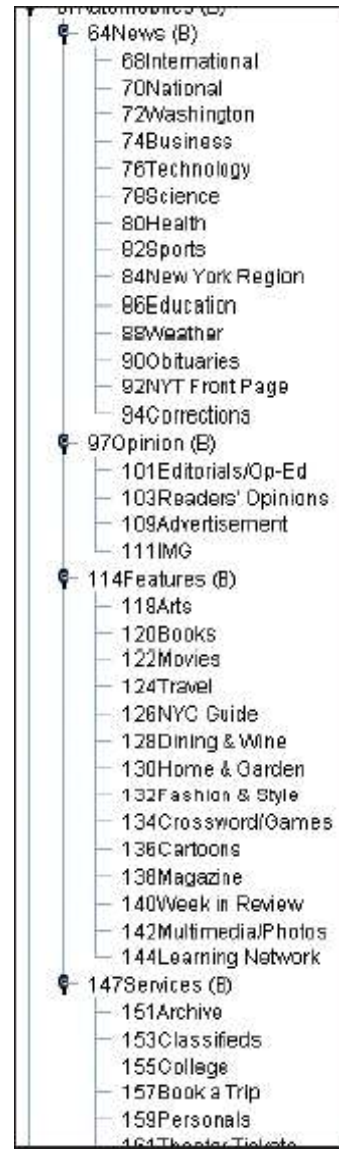


Abbildung 3b: Ziel: fertige Taxonomie

Um zu diesem Ziel zu gelangen, werden drei Verfahren nacheinander angewandt: Binäres Semantisches Partitionieren, Gruppieren und Befördern. Es folgt nun die Erklärung dieser Verfahren:

Das Ziel des Binären Semantischen Partitionierers ist es, aus dem Syntaxbaum der ursprünglichen Webseite einen Binärbaum zu generieren, in dem möglichst ähnliche Knoten des Ursprungsbaumes zusammengefasst werden. Dies wird mittels eines dynamischen Algorithmus' erreicht, der für alle möglichen Gruppierungen zweier Knoten den Aufwand bestimmt, der für diese Gruppierung nötig ist; dieser Aufwand gibt den Grad an „Unähnlichkeit“ zweier Knoten an und berechnet sich aus verschiedenen Kostenfaktoren, auf die ich hier nicht im Detail eingehen werde.

Zwischenergebnis: Abbildung 4

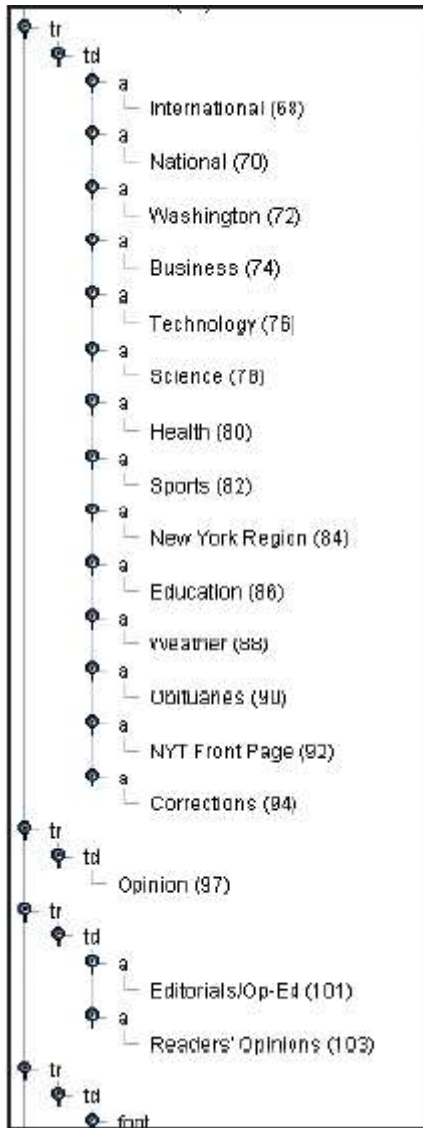


Abbildung 4a: Ursprünglicher XML-Baum



Abbildung 4b: gewonnener Binärbaum

Im nächsten Schritt wird nun der Binärbaum aufgelöst und sein Inhalt in zusammengehörige Gruppen verschmolzen. Dazu wird zu Beginn ein Grenzwert d festgelegt und alle Blattknoten werden als „einfach“ markiert. Danach wird der Baum im Post-Order-Verfahren durchlaufen und dabei folgendes vorgenommen:

- Werden zwei „einfache“ Geschwisterknoten gefunden und ist der Aufwand, diese zu gruppieren, kleiner als d , so werden diese Blattknoten als „Instanzen“ und ihr Elternknoten als „Gruppenknoten“ markiert.
- Werden zwei Geschwisterknoten gefunden, die beide als Gruppenknoten markiert wurden, und ist der Aufwand, diese zu gruppieren, kleiner als d , so wird deren Elternknoten als Gruppenknoten markiert und ihre Instanzen werden miteinander verschmolzen.
- Tritt der Fall auf, dass einer der beiden Geschwisterknoten ein einfacher und einer ein Gruppenknoten ist, und beträgt der Aufwand, den einfachen Knoten mit den Instanzen des Gruppenknotens zu verschmelzen, weniger als d , so wird der Typ des einfachen Knotens in Instanz geändert und mit den Instanzen des Geschwisterknotens (der ja ein Gruppenknoten ist) verschmolzen.

Der Algorithmus stoppt schließlich im Wurzelknoten.

Nun sind also alle Blattknoten in Gruppen eingeteilt, sie sind Instanzen ihrer jeweiligen Gruppen.

Das Zwischenergebnis nach diesem Schritt ist in Abbildung 5 dargestellt.

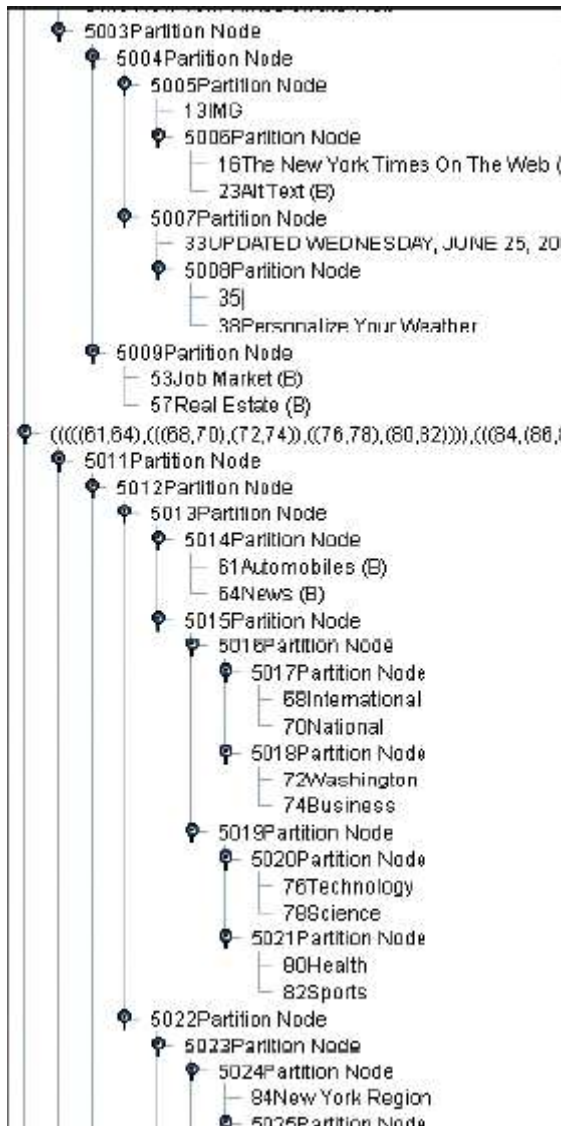


Abbildung 5a: Binärbaum

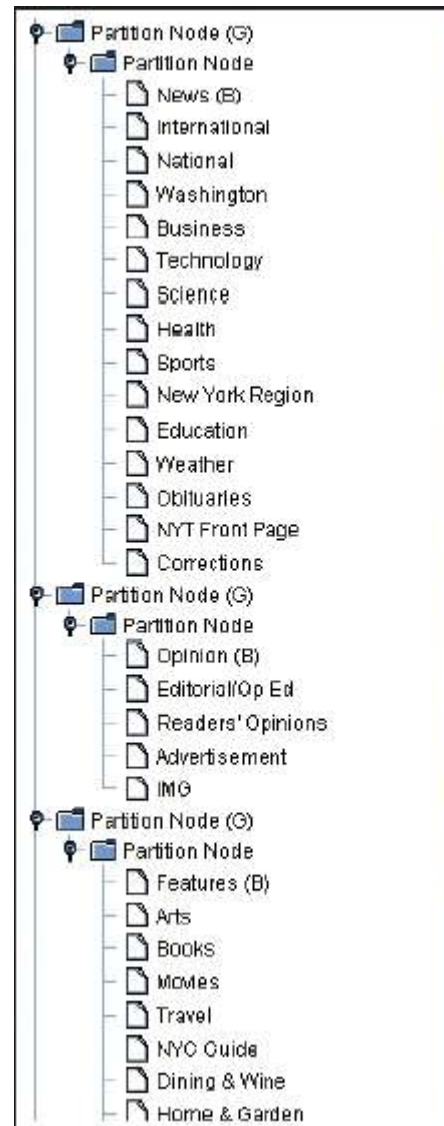


Abbildung 5b: „Gruppenbaum“

Zu sehen ist, dass die Struktur des Baumes inzwischen wesentlich einfacher ist. Nun stecken aber noch immer sämtliche Informationen in den Blattknoten und es existiert noch keine Hierarchie, da die Gruppenknoten nicht benannt sind. Deshalb werden nun abschließend bestimmte Blattknoten ausgewählt und durch Ersetzen der „Partition Nodes“ ihren Geschwisterknoten übergeordnet, damit die erwünschte Hierarchie hergestellt wird.

Ein Knoten, der dafür in Betracht gezogen wird, übergeordnet zu werden, muss einer der beiden folgenden Regeln folgen:

1. der Knoten ist das erste Kind seines Elternknotens, der Elternknoten ist nicht als „Gruppenknoten“ markiert und der Knoten ist „betont“, d.h. er erfüllt eine dieser Bedingungen:
 - es gibt einen HTML-Tag im Pfad des Knotens, der ihn „bold“ also „fettgedruckt“ macht, wie z.B. , <bold> oder <h1>
 - es gibt einen solchen „bold“-Tag im „class“-Attribut des Knotens (also wie die vorige Bedingung, nur dass die Definition des „bold“-Attributs an anderer Stelle steht) (s. auch Cascading Style Sheets in der HTML-Literatur)
 - der Text des Knotens ist vollständig großgeschrieben
 - es gibt einen - oder -Tag im Pfad zum darauffolgenden Knoten (d.h. Der nächste Knoten ist Teil einer Auflistung)
2. der Knoten muss eine der folgenden Bedingungen erfüllen:
 - er ist das erste Kind seines Elternknotens (da meistens z.B. die Überschrift einer Navigationsleiste den Inhalt der darunterliegenden HTML-Links beschreibt)
 - sein Elternknoten ist nicht als Gruppenknoten markiert
 - er ist einzeln oder sein einziger Geschwisterknoten ist ein Gruppenknoten

Während der Beförderung ersetzt ein Knoten, der befördert wird, seinen Elternknoten. Falls die Regel zur Beförderung wieder zutrifft, wird der „betonte“ Knoten noch einmal befördert.

Das Ergebnis dieses Schrittes ist in Abb. 6 zu sehen (links vor, rechts nach dem Befördern):

Zu sehen ist, dass z.B. „News“ und „Opinion“ an die Stellen der „Partition Nodes“ gewandert und ihre ehemaligen Geschwisterknoten ihnen nun untergeordnet sind. Hier sind Knoten, die als „bold“ markiert sind, mit (B), Knoten, die als „Gruppenknoten“ markiert sind, mit (G) bezeichnet.

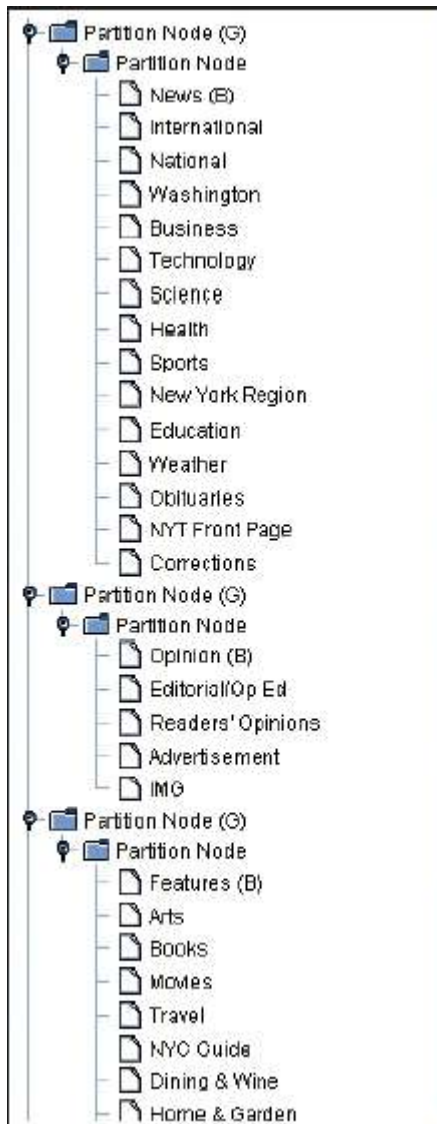


Abbildung 6a: "Gruppenbaum"

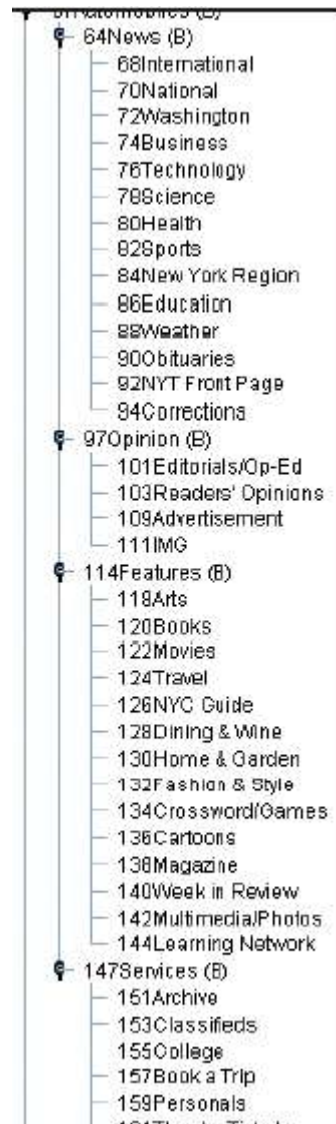


Abbildung 6b: Baum nach dem Beförderungsschritt

3. Taxonomy Mining

In den bisherigen Abschnitten wurde gezeigt, wie eine einzelne gegebene Webseite von OntoMiner in einen Baum überführt wird. Die Bäume, die aus mehreren sich inhaltlich überlappenden Seiten generiert wurden, werden vom Taxonomy Miner dazu verwendet, automatisch Taxonomien zu erstellen.

Im Verlauf des Verfahrens werden Konzepte (die Oberbegriffe bzw. Kategorien, die wir mit Inhalt füllen wollen) von Instanzen (die Inhalte für die einzelnen Konzepte) getrennt, sowie von „Dekoration“, die für die Darstellung für den Menschen benötigt wird, wie z.B. Navigationsleisten.

Zudem werden „is-a“-Beziehungen zwischen den Konzepten herausgearbeitet, also Hierarchien, bei denen ein Konzept eine Spezialisierung eines anderen Konzepts ist (vergleiche auch Vererbung bei der objektorientierten Programmierung).

In den folgenden Abschnitten werde ich nun erklären, welche Schritte dazu durchlaufen werden.

3.1 Häufigkeitsbasiertes Mining

Die Idee hinter diesem Verfahren ist, dass wichtige Labels (Bezeichnungen und damit Inhalt der Knoten) innerhalb einer Seite häufig wiederholt werden. Nachdem eine Seite zuerst mittels Semantischer Partitionierung vorverarbeitet und somit in ein halbstrukturiertes XML-Dokument umgewandelt wurde, werden nun die häufigen Labels extrahiert. Als häufig bezeichnet man die Labels, die mindestens so oft auftreten wie ein zuvor festgelegter Grenzwert angibt (ein geeigneter Grenzwert wurde zuvor experimentell bestimmt).

3.2 Extraktion weiterer in Frager kommender Labels

In der vorherigen Phase haben wir alle Labels identifiziert, die häufig vorkommen. Nun ist es aber möglich, dass Labels auftauchen, die zwar wichtig sind, aber nicht oft auftreten. Um solche Labels zu finden, werden im HTML-Baum die Pfade zu den häufigen Labels ausgelesen und alle anderen Labels als wichtig eingestuft, die zwar nicht häufig sind, aber die selben Attribute wie häufige Labels haben.

3.3 Kürzungsphase

Es ist möglich, dass wir in der letzten Phase einige inhaltlich unwichtige Labels extrahiert haben (etwa den Link zum Shop der New York Times im Beispiel). Um diese wieder zu beseitigen, ignorieren wir Labels, die eine der folgenden Bedingungen erfüllen:

- sie enthalten keine URL
- die URL verweist auf eine andere Domain
- die Seiten, auf die sie jeweils verweisen, enthalten keine häufigen Labels oder gültigen Instanzen (Erklärung für letzteres folgt in Abschnitt 4).

3.4 Gruppieren der Labels in Konzepte

In den bisherigen Phasen haben wir die wichtigen Labels gesammelt. Da gleiche Begriffe auf verschiedenen Seiten oftmals unterschiedlich aussehen (z.B. „sport“ und „sports“) werden bis hier redundante Labels existieren. Um diese Redundanz zu beseitigen werden die Begriffe nun entsprechend ihrer lexikographischen Ähnlichkeit gruppiert. Diese entstehenden Gruppen werden Konzepte genannt. Also ist beispielsweise „Sport“ ein Konzept; dieses wird sowohl über „sport“ als auch „sports“ identifiziert.

Beachte, dass OntoMiner im jetzigen Entwicklungsstand nur lexikographische, nicht jedoch semantische Ähnlichkeit erkennt.

3.5 Herausarbeiten von „is-a“-Beziehungen aus hierarchisch partitionierten Webseiten

Die Ergebnisse der vorhergehenden Gruppierungsphase sind „flach“. Um eine Taxonomie abzuleiten, müssen „is-a“-Beziehungen zwischen ihnen herausgearbeitet werden.

Ein Algorithmus findet nun aus dem semantisch partitionierten Baum die Eltern-Kind-Beziehungen und diejenigen höheren Grades (Großeltern-Kind-Beziehungen usw.) und erstellt aus den vorher erarbeiteten Sammlungen von Konzepten die Konzept-Taxonomie zu den gewünschten Web-Seiten.

3.6 Erweitern der Taxonomie auf gesamte Websites

Die Taxonomie, die wir im vorherigen Schritt erhalten haben, entspricht der, die wir aus den Startseiten der Websites herausgearbeitet haben.

Um die Taxonomie des gesamten Fachgebiets zu erhalten, folgen wir nun den Links, die zu jedem der einzelnen Konzepte gehören, und wiederholen die vorherigen Schritte mit jeder Seite. Indem wir jedes Ergebnis seinem „Vorgänger“ unterordnen, erweitern wir die Taxonomie in die Tiefe.

Beispielsweise haben wir bisher „Sports“ als Konzept erhalten. Folgen wir jetzt allen Links, die mit „Sports“ zu tun haben, so ergeben sich Unter-Taxonomien wie z.B. „Baseball“, „Tennis“ und „Pferderennen“.

Die Gesamtheit dieser Taxonomien ergibt die endgültige Taxonomie für dieses Fachgebiet.

4. Instanzenextraktion

Ein Taxonomie-Schema besteht aus einer Menge von „is-a“-Relationen zwischen Konzepten. Um eine Taxonomie zu „füllen“ muss der OntoMiner die Konzept-Instanzen identifizieren, also sozusagen die Vertreter eines Konzepts.

4.1 Identifizieren des Instanzsegments

Viele Webseiten zeigen den Inhalt zu einem Begriff, den man beispielsweise durch Anklicken eines Links ausgewählt hat, nur als Teil der geladenen Seite an. Der Rest ist gefüllt mit Navigationsleisten, Werbung, Hinweisen auf völlig andere Inhalte usw.

Nun versucht OntoMiner, denjenigen Block zu finden, der den gewünschten Inhalt hat. Daraus werden dann die Instanzen ausgelesen und in die Taxonomie, die der Taxonomy Miner erstellt hat, eingefügt. Hierzu werden zwei ähnliche Links aufgerufen und mit dem Flat Partitioner in Blöcke aufgeteilt. Danach werden die einander entsprechenden Blöcke aus den beiden Seiten verglichen. Ist der Inhalt eines Blocks aus Seite 1 gleich oder sehr ähnlich zu dem entsprechenden in Seite 2, so werden diese Blöcke als Kopf- oder Fusszeile, Werbung oder Navigationsleiste gesehen; sind sie jedoch verschieden, so geht man davon aus, dass es sich um den Inhalt handelt, der die gesuchten Instanzen enthält. Deshalb wird dieser Block Instanzsegment genannt.

Ist diese Verarbeitung durchgeführt, so betrachtet das Programm nun die gemeinsamen Teile der Pfade zu den Instanzen im XML-Baum. Diese werden als Signaturpfade bezeichnet. Jede Seite, die Signaturpfade enthalten, nennt man „template-driven“ (man kann gedanklich eine Schablone darüberlegen und erhält daraus die Instanzinformationen, der Aufbau der Seiten, die „template-driven“ sind, ist also ähnlich).

Bevor die Instanzinformationen aus diesen Seiten ausgelesen werden, wird für jeden Block das Verhältnis der Links, die zu Seiten führen, die „template-driven“ sind, zur Gesamtzahl der Links in diesem Block berechnet. Das Segment, für das dieses Verhältnis am größten ist, wird als Hauptinstanzsegment (HIS) bezeichnet.

Die Links auf der Konzeptseite (also z.B. dem Inhaltsverzeichnis) werden aufgerufen und die Zielseiten werden jeweils überprüft, ob sie Signaturpfade enthalten. Wenn dies der Fall ist, werden die damit gefundenen Instanzen ausgelesen.

4.2 Extrahieren von Attributen und Werten aus Instanzsegmenten

Nachdem das HIS gefunden wurde, ruft der OntoMiner den Hierarchischen Partitionierer auf, der die hierarchischen Strukturen zu all denjenigen Webseiten erstellt, auf die vom HIS aus verwiesen wird.

Danach wird die Hierarchie zwischen den häufigen Labels in diesen Strukturen erstellt. Dabei werden häufige Labels als Attribute, ihre Kindknoten als Werte bezeichnet. Die Attribute und die Beziehungen zwischen ihnen werden zusammen mit der Hierarchie gespeichert.

Einige der Dateninstanzen haben möglicherweise keine häufigen Labels über sich. In diesen Fällen werden solche Datenwerte durch ihre Signaturpfade identifiziert, es wird also eine Tabelle mit Attributen geführt, wobei die Spalten die Pfade und die Zeilen die Instanzsegmente beinhalten. Die Entwickler planen, falls Attributnamen fehlen, diese automatisch aus anderen Attributen zu erschließen.

Damit sind alle verwendeten Verfahren vorgestellt. Nun folgt eine Betrachtung der Ergebnisse:

5. Experimentell erlangte Ergebnisse

Die Forscher haben ihr Programm mit verschiedenen Webseiten aus unterschiedlichen Fachbereichen getestet, unter anderem mit Nachrichten-, Biologie- und Hotelseiten.

Zuerst wird manuell eine Hierarchie erstellt, die als Referenz betrachtet wird.

Um die Qualität der erhaltenen Taxonomien beurteilen zu können, werden nun durch Vergleich mit dem Referenzbaum zwei Werte berechnet: Precision und Recall.

Der Precision-Wert berechnet sich aus dem Verhältnis *richtig* erkannter Eltern-Kind-Beziehungen (auch Großeltern-Enkel-Beziehungen usw.) zur *Gesamtzahl* an erkannten Beziehungen, während der Recall-Wert das Verhältnis von *richtig* erkannten Beziehungen zu der im Ursprungsbaum *tatsächlich vorhandenen* Beziehungen darstellt.

Anders ausgedrückt ist also Precision ein Maß dafür, wieviele der erkannten Beziehungen auch im Referenzbaum vorhanden sind, und Recall dafür, wieviele der Beziehungen, die im Referenzbaum vorhanden sind, auch tatsächlich in die Taxonomie aufgenommen werden.

Die Werte für Precision und Recall schwanken je nach untersuchtem Fachgebiet im Allgemeinen zwischen 75% und 96%. Durch Ändern verschiedener Parameter wie z.B. den Grenzwerten kann die Genauigkeit oftmals noch erhöht werden.

Im einzelnen ergeben sich für die verschiedenen Verfahren folgende Werte:

- Der Hierarchische Partitionierer erreicht Durchschnittswerte von 87% Precision und 82% Recall.
- Beim Taxonomy Mining wird eine Precision von 91% und ein Recall von 79% erzielt.
- Bei der Instanzenextraktion ist je nach Phase eine Precision von 80%-85% und ein Recall von 84%-97% möglich.

6. Bewertung

Dieses Proseminar steht unter dem Titel „Künstliche Intelligenz“. Von diesem Standpunkt aus betrachtet ist der OntoMiner nicht sehr weit fortgeschritten. Von künstlicher Intelligenz könnte man sprechen, wenn das Programm die Semantik der Webseiten berücksichtigen würde, anstatt ausschließlich auf den syntaktischen Besonderheiten von Webseiten aufzubauen.

Die Zielsetzung der Unterstützung des Semantic Web erscheint mir jedoch sehr erstrebenswert und dieses Projekt hat sicherlich seine Berechtigung, da es bereits gute Ergebnisse erzielt. Um in der Praxis interessant zu sein, sollten die Durchschnittswerte für Precision und Recall aber noch nahe an die 100%-Marke steigen, damit davon auszugehen ist, dass die Ergebnisse sinnvoll sind.

7. Ausblick: Was ist heute möglich ?

Die britische Firma Autonomy ist der Hersteller der gleichnamigen Software, die es bereits heute ermöglicht, Informationen nicht nur syntaktisch, sondern auch inhaltlich zu analysieren. So ist es beispielsweise folgendes Vorgehen möglich: Der Anwender gibt als Suchbegriff „Nahost-Politik“ ein. Das Programm findet nicht nur Dokumente, in denen dieser Begriff lexikalisch vorkommt, sondern auch z.B. eine Fernsehmeldung über den Bericht eines Waffenkontrolleure, der im Irak nach Atomwaffen sucht. Dabei ist die Software nicht einmal auf ein bestimmtes Medium beschränkt, sondern findet sowohl Texte als auch Video- oder Audio-Aufzeichnungen.

Eine andere -allerdings noch fiktive- Anwendung ist die, Firmen-E-Mails automatisch darauf zu untersuchen, ob ihr Inhalt eine schädliche Wirkung auf die Firma haben könnte und den Benutzer schon vor dem Abschicken darauf hinzuweisen.

Datenschützer sehen in solcher Software allerdings eine immense Gefahr: Eine totale Überwachung, durch die Informationen, die vom Kontrollierenden (Staat/Konzern) nicht erwünscht sind, nicht übermittelt werden dürfen. Dies bringt das Szenario aus George Orwell's Buch „1984“ erschreckend nahe – und das praktisch ohne personellen Aufwand und wesentlich weniger utopisch als in der fiktiven Erzählung.

8. Literatur:

- [1] Hasan Davulcu, Srinivas Vadrevu, Saravanakumar Nagarajan. OntoMiner: Bootstrapping Ontologies From Overlapping Domain Specific Web sites. IEEE Intelligent Systems no.5, Sept./Okt.2003, S.24-33
- [2] Wikipedia Online-Enzyklopädie, <http://de.wikipedia.org>
- [3] Peter Schüler. Detektei Allwissend. Heise Verlag, c't 8/2004, S.86ff