Content Extraction: Bestimmung des Hauptinhaltes in HTML Dokumenten

Thomas Gottron

Institut für Informatik
Johannes Gutenberg-Universität
55099 Mainz
gottron@uni-mainz.de

Abstract: Außer dem Artikel der den eigentlichen Hauptinhalt darstellt enthalten die meisten HTML Dokumente im WWW zusätzliche Inhalte, wie beispielsweise Navigationsmenüs, gestalterische Elemente oder Werbung. Für verschiedene Anwendungen ist es nötig die Unterscheidung zwischen Haupt- und zusätzlichen Inhalten automatisch vorzunehmen. Content Extraction und Template Detection sind Verfahren, die diese Aufgabe lösen. Während der Forschungsarbeit auf diesem Gebiet sind einige interessante Beiträge entstanden. Drei davon sollen hier kurz vorgestellt werden. Dazu gehört der neu eingeführte Content Code Blurring Algorithmus, derzeit der leistungsfähigste Ansatz zur Inhaltsextraktion. Der zweite Beitrag liegt in der Entwicklung objektiver Maße zur Bewertung der Leistung von Algorithmen zur Inhaltsextraktion. Dadurch ließen sich bestehende Verfahren erstmals überhaupt miteinander vergleichen. Eine Analyse verschiedener Methoden zur Gruppierung von Webdokumenten bezüglich der ihnen unterliegenden Templates stellt den dritten größeren Beitrag dieser Arbeit dar. In Kombination mit einer lokalen Websuche kann dieses Templateclustering für die automatische Erstellung von Trainingsdatensätzen zur Templateerkennung eingesetzt werden. Da das Verfahren vollautomatisch ablaufen kann, ermöglicht es im Prinzip Template Detection auf einzelne Dokumente anzuwenden. Damit lassen sich die Vorteile aus Content Extraction und Template Detection verknüpfen.

1 Einführung

Content Extraction beschäftigt sich, vereinfacht gesagt, mit algorithmischen Lösungen, um in Webdokumenten den Hauptinhalt zu identifizieren und zu extrahieren [Got08d]. Der Hintergrund ist folgender: wenn man sich heutzutage eine Seite im Web anschaut, dann stellt man schnell fest, dass der eigentliche, für den Nutzer interessante Hauptinhalt von vielen zusätzlichen Inhalten umgeben ist. Navigationsmenüs, Werbung, Copyright Vermerke, Verweise auf ähnliche Inhalte oder gestalterische und visuelle Elemente sind typische Beispiele für diese zusätzlichen Inhalte. Uns Menschen erschließt sich in der Regel recht schnell, auf welchen Teil des gesamten Dokumentes wir unsere Aufmerksamkeit lenken müssen, so dass wir wie in Abbildung 1 angedeutet den Hauptinhalt recht schnell isolieren können. Für einen Computer jedoch, ist das Erkennen der wichtigen Inhalte ein äußerst schwieriges Unterfangen, nicht zuletzt weil das Verständnis der Inhalte fehlt.

Abbildung 1: Der Hauptinhalt in einem Webdokument

Dennoch ist diese Aufgabenstellung in vielen Anwendungen von praktischer Relevanz. Eine automatische Bestimmung des Hauptinhaltes in einem Webdokument ist immer dann gefragt, wenn ein Programm den Menschen beim Lesen oder Verstehen von Webdokumenten oder der Suche nach Informationen unterstützen soll. Das sind klassische Aufgaben im Bereich des Information Retrieval und des Web Content Mining, wie man sie aus den alltäglich gewordenen Websuchmaschinen kennt. Dazu gehört aber auch die Inhaltsanalyse, beispielsweise zur Auswertung des Stimmungsbildes unter Forennutzern oder zur Bestimmung der Lesbarkeit und Nutzerfreundlichkeit. Ein weiteres Einsatzgebiet ist die Verbesserung der Zugänglichkeit von Webdokumenten über nicht-Standard Webbrowser (z.B. Screenreader für Sehbehinderte oder auf Geräten mit kleinem Bildschirm).

Algorithmische Lösungen dieser Aufgabe lassen sich auf oberster Ebene in zwei Klassen aufteilen: Content Extraction Verfahren und Template Detection Ansätze. Die Abgrenzung der beiden Verfahren liegt in ihrer Herangehensweise und der verwendeten Datenbasis.

Content Extraction (CE) Verfahren betrachten nur ein einzelnes Dokument. Sie formalisieren gewisse Annahmen über Form, Art und Eigenschaften des Hauptinhaltes in HTML Dokumenten. Mit sehr unterschiedlichen, teilweise heuristischen Verfahren wird dann versucht, jene Stellen im Dokument zu identifizieren, die den Annahmen über den Hauptinhalt am ehesten entsprechen.

Template Detection (TD) Verfahren gehen anders vor. Sie nutzen die immer stärkere Verbreitung von Content Management Systemen und deren Templatemechanismen aus. Da die Dokumente eines Webauftrittes größtenteils auf einigen wenigen Vorlagen beruhen, ist die Idee, aus einer Menge von Trainingsdokumenten die Form dieser Templateschablone abzuleiten. Damit lässt sich dann die Position des Hauptinhaltes in den Dokumenten abschätzen und dies kann für eine Extraktion genutzt werden.

Auf einige neue Erkenntnisse in diesen beiden Themengebieten wird nachfolgend etwas näher eingegangen. Die jeweiligen Neuerungen im Bereich der CE Algorithmen, der Evaluation von Inhaltsextraktion und der Erzeugung von Trainingsmengen für TD Ansätze werden dabei möglichst anschaulich dargestellt. Für technische Details sind aber jeweils die Originalpublikationen angegeben.

2 Ein neuer Content Extraction Ansatz

Algorithmische Ansätze zur Bestimmung des Hauptinhaltes finden sich in der einschlägigen Literatur einige. Meist vor dem Hintergund der Datenbereinigung für Text Mining oder Information Retrieval Verfahren, zur Verbesserung der Nutzerinteraktion auf Geräten mit kleinem Bildschirm (Mobiltelefone) oder zur Unterstützung von sehbehinderten Nutzern.

Eines dieser Verfahren ist Body Text Extraction (BTE) [FKS01]. Die Idee in BTE ist, den Quelltext eines HTML Dokumentes in eine Sequenz von Tags und Wörter zu tokenisieren und in dieser Sequenz einen Anfangs- und Endpunkt des Hauptinhaltes zu bestimmen. Die Position der beiden Punkte wird durch die Bestrebung festgelegt, die Anzahl der enthaltenen Worttoken und ausgeschlossenen Tagtoken zu maximieren. Die Nachteile, dass damit nur durchgängige Abschnitte im Text bestimmt werden können und dass der Optimierungsprozess recht aufwändig ist werden im DSC Ansatz [PBC$^+$02] behandelt. Das Verfahren bestimmt in der Tokensequenz solche Abschnitte die im Mittel deutlich weniger Tagtoken enthalten als das gesamte Dokumente im Durchschnitt.

Anders als diese Verfahren setzen Link Quota Filter (LQF), wie z.B. in [MOC05] beschrieben, nicht direkt auf dem Quellcode auf, sondern eine Ebene höher auf dem daraus erzeugbaren DOM-Baum. Hier wird in Teilbäumen, die beispielsweise Absätzen oder Tabelleneinträgen im Dokument entsprechen, das Verhältnis von Hyperlinkankern (d.h. Texten in `a`-Elementen) zu normalen Texten bestimmt. Liegt dieses Verhältnis über einem vorgegebenen Schwellwert, so stellt dieser Dokumentblock höchstwahrscheinlich einen Teil des Navigationsmenüs oder einer Liste von Referenzen dar. Diese gehören in den seltensten Fällen zum Hauptinhalt und werden daher aus dem Dokument herausgefiltert. Ein solcher Filter ist auch Bestandteil des von Gupta et al. [GKNG03] vorgestellten Crunch-Systems. Über LQF hinaus werden in Crunch auch einige andere Heuristiken angewendet, um Listenstrukturen und Werbebanner zu erkennen.

Einen anderen Ansatz verfolgt das neu entwickelte Content Code Blurring (CCB) [Got08c]. Textuelle Hauptinhalte sind typischerweise länger und gleichmäßig formatiert während zusätzliche Inhalte meist kurz und stark formatiert sind. Diese Eigenschaften soll ausgenutzt werden. Da die Festlegung des Layouts oder der Struktur mit Tags im Quellcode gleichzusetzen ist, sucht CCB im Quellcode nach Regionen mit viel Inhalt und wenig Code.

Dazu wird der Quellcode zunächst in atomare Elemente unterteilt, die eindeutig Textinhalten oder Codeelementen zuordenbar sind. Für diese atomaren Elemente wird dann bestimmt, ob ihre Nachbarschaft vorwiegend aus Code- oder Inhaltselementen besteht. In der Basisvariante des Algorithmus wird dabei für jedes einzelne Zeichen im Quellcode

Abbildung 2: Grafische Interpretation des CCB Verfahrens

bestimmt, wie sehr es von Inhalt oder Code umgeben ist. Die TCCB (Token-CCB) Form des Algorithmus setzt wie BTE und DSC auf Wort- und Tag-Token auf.

Das Verhältnis von Code zu Inhalt in der Nachbarschaft eines Elementes wird über lokale und gewichtete Mittelwerte bestimmt. Näher gelegene Elemente haben einen stärkeren Einfluss als weit entfernte Element. Dies lässt sich dadurch modellieren, dass die Gewichte gemäß einer Gaussverteilung besetzt werden, wobei der Erwartungswert jeweils über dem gerade betrachteten Element liegt. Das läßt auch eine sehr schöne visuelle Interpretation zu, die letztendlich namensgebend ist und in Abbildung 2 verdeutlicht wird.

Stellt man sich die Abfolge von Elementen als eindimensionalen Bild vor, in dem jedes Inhaltselement als ein weißer Punkt und jedes Codeelement als schwarzer Punkt dargestellt wird, so entspricht die Berechnung der lokalen Gauss-gewichteten Mittelwerte einem Gauss'schen Weichzeichnen (Blurring) wie man es aus Bildbearbeitungsprogrammen kennt. Wiederholt man das Weichzeichnen einige Male so werden im Bild gleichmäßige helle und dunkle Regionen sichtbar. Helle Regionen stehen im übertragenen Sinne für inhaltsreiche Dokumentabschnitte (hier waren anfangs viele weiße Inhaltspixel) und dunkle Regionen repräsentiert Dokumentfragmente die eher von Code dominiert werden (hier waren anfangs schwarze Codepixel vorherrschend). Vereinfacht gesagt werden nun die helleren Regionen als Hauptinhalt extrahiert. Dazu wird ein Schwellwert für die benötigte Helligkeit angelegt und nachfolgend noch solche Textpassagen an den Übergängen komplettiert, die eventuell nur teilweise extrahiert wurden.

Die Erfahrung mit anderen Verfahren hat gezeigt, dass stark von Hyperlinks durchzogene Hauptinhalte generell schwierig zu detektieren sind. Daher wurde auch eine Variante (ACCB) entwickelt, die durch Links bedingte Codeelemente in der Berechnung der lokalen Mittelwerte ignoriert. Das könnte zwar zu Schwächen in der Erkennung von durch Linkstrukturen charakterisierten Navigationslementen führen, diese werden jedoch – wie man an den Ergebnissen im nächsten Abschnitt sehen kann – von den Vorteilen mehr als aufgewogen, so dass dieses Vorgehen für CCB generell empfehlenswert ist.

3 Evaluation von CE Verfahren

Eine zentrale Frage bei der Entwicklung oder beim Einsatz von CE Verfahren ist deren Performanz. Dabei steht meist weniger die Effizienz als die Effektivität im Blickpunkt.

Es stellt sich also die Frage wie exakt ein Verfahren den tatsächlichen Hauptinhalt in einem Dokument bestimmen kann. Ein Evaluationsansatz besteht darin, einen menschlichen Nutzer mit der Beurteilung zu beauftragen (so z.B. in [CMZ03]). Das ist jedoch recht mühsam, fehleranfällig und nicht automatisierbar. Ein anderer, ebenfalls häufig angewendeter Ansatz ist, die CE Verfahren direkt nach ihrer Auswirkung auf einen bestimmten Anwendungshintergrund zu beurteilen. So wird teilweise die Auswirkung auf Anwendung zur Text Klassifikation oder im Information Retrieval betrachtet. Gupta [GKNG03] vergleicht hingegen, wie viel schneller ein Screenreader ein Webdokument vorlesen kann, nachdem es durch das Crunch System gefiltert wurde. Solche Evaluationsverfahren erlauben es nur schwer direkte Aussagen über die Qualität des Algorithmus zu machen. Liefert ein Verfahren beispielsweise immer nur ein Wort als Hauptinhalt, lässt sich dieses extrem schnell von einem Screenreader vorlesen – dies dürfte im Allgemeinen aber kaum einem extrem guten Auffinden des Hauptinhaltes entsprechen.

Der Ansatz der hier vorgestellt wird setzt auf eine direkte Bewertung der Inhaltsextraktion, die nach einer anfänglichen Erstellung einer Testkollektion zudem vollautomatische ablaufen kann [Got07].

Dazu benötigt man eine Sammlung von Testdokumenten, zu denen der Hauptinhalt bekannt ist. Die Erzeugung eines solchen Goldstandards kann manuell geschehen: die Dokumente werden einem Nutzer gezeigt und dieser markiert den Teil des Dokumentes den er für den Hauptinhalt hält. Alternativ kann man jedoch auch automatisch vorgehen, indem man zu Dokumenten die auf einem bekannten Template aufsetzen ein speziell zugeschnittenes Extraktionsprogramm für den Hauptinhalt schreibt. Damit kann man dann massiv Testdaten aus allen Dokumenten erzeugen, die auf diesem Template basieren. Unter Anwendung dieser beiden, insbesondere der letzten Vorgehensweisen konnte so eine sehr große Kollektion von über 9.000 Testdokumenten erzeugt werden.

Für die Evaluation eines CE Verfahrens wird dieses nun mit den Testdokumenten konfrontiert. Den Text, den das Verfahren als Hauptinhalt identifiziert wird anschließend mit dem Goldstandard verglichen. Eine der Möglichkeiten diesen Vergleich durchzuführen besteht darin, beide Texte als Folge von Wörtern aufzufassen und darin die längste gemeinsame Teilfolge (longest common subsequence) von Wörtern zu bestimmen. Vergleicht man nun die Länge dieser gemeinsamen Teilfolge mit der Länge der Wortsequenz im Goldstandard so erhält man den Anteil des Hauptinhaltes, der korrekt identifiziert wurde. Dies entspricht der Auffassung von Recall im Information Retrieval. Der Vergleich der gemeinsamen Teilsequenz mit dem vom CE Verfahren gelieferten Text ergibt den Anteil des extrahierten Textes, der tatsächlich zum Hauptinhalt gehört. Das entspricht im Information Retrieval dem Konzept der Precision. Recall und Precision lassen sich zu einem gemeinsamen Maß verrechnen, dem sogenannten F1 Maß [VR79]. Dieses nimmt Werte zwischen 0 und 1 an. In unserem konkreten Fall liegen die Werte um so näher an 1 liegen, je präziser der Hauptinhalt ermittelt wurde. Beide Arten der Abweichung – Extraktion von zu viel oder zu wenig Text – werden bestraft. In Tabelle 1 wird die Performanz der oben angesprochenen Verfahren bezüglich dieses F1 Maßes aufgelistet.

Bei der Untersuchung der verschiedenen Algorithmen erwies sich die ACCB Variante unseres oben beschriebenen Algorithmus als die mit der besten Leistung bezüglich des F1 Maßes. Dies gilt sowohl im globalen Durchschnitt, als auch bei Betrachtung der einzelnen

Tabelle 1: F1-Ergebnisse verschiedener CE Verfahren

	bbc	chip	economist	espresso	golem	heise	manual	repubblica	slashdot	spiegel	telepolis	wiki	yahoo	zdf	Macro Avg.	Micro Avg.
accb	0.924	0.703	0.890	0.875	0.959	0.916	0.419	0.968	0.177	0.861	0.908	0.682	0.847	0.929	0.790	0.848
bte	0.676	0.262	0.736	0.835	0.532	0.674	0.409	0.842	0.113	0.749	0.927	0.853	0.602	0.875	0.649	0.697
ccb	0.923	0.716	0.914	0.876	0.939	0.841	0.420	0.964	0.160	0.858	0.913	0.403	0.742	0.929	0.757	0.799
crunch	0.756	0.342	0.815	0.810	0.837	0.810	0.382	0.887	0.123	0.706	0.910	0.725	0.738	0.772	0.687	0.750
dsc	0.937	0.708	0.881	0.862	0.958	0.877	0.403	0.925	0.252	0.902	0.859	0.594	0.780	0.847	0.770	0.823
lqf	0.834	0.502	0.732	0.667	0.926	0.791	0.387	0.826	0.135	0.790	0.906	0.690	0.799	0.579	0.683	0.764
tccb	0.914	0.842	0.903	0.871	0.947	0.821	0.404	0.918	0.269	0.910	0.902	0.660	0.873	0.745	0.784	0.837

Testszenarien.

Außer der reinen Extraktionsperformanz können noch andere Indikatoren bestimmt werden. Dazu gehört einerseits die Laufzeit, die von Interesse ist, wenn die Dokumente beispielsweise direkt beim Abruf aus dem Web on-the-fly analysiert werden sollen. Nimmt man den BTE Algorithmus aus, laufen alle angesprochenen Verfahren ausreichend schnell. Selbst bei längeren Dokumente spürt ein Nutzer selten ein Verzögerung von mehr als einer Sekunde.

Ebenfalls interessant ist die Stabilität eines Algorithmus. Das heißt, dass man bei ähnlichen Dokumenten eine ähnliche Qualität bezüglich der Extraktionsleistung erwarten kann. Letzteres lässt sich sehr gut auf den automatisch erzeugten Testdaten bestimmen, da hier immer sehr ähnliche Dokumentstrukturen vorliegen. Ein Schätzer für die Varianz der Extraktionsleistung auf diesen Testpaketen gibt Auskunft darüber wie unterschiedlich die Ergebnisse ausfallen können. Auch hier gibt es unter den angeführten Algorithmen kaum Unterschiede; der ACCB Ansatz liegt jedoch leicht vor den anderen Verfahren.

4 Template Clusterung zur Erzeugung von Trainingsdaten

Anders als CE Verfahren, die isoliert auf einem einzelnen Dokument arbeiten gehen TD Ansätze von mehreren Dokumenten aus, in denen sie Templatesturkturen erkennen wollen. Der erste Algorithmus auf diesem Gebiet wurde von Bar-Yossef und Rajagopalan [BYR02] vorgestellt. Darin werden die Dokumente zunächst in Blockstrukturen segmentiert (sog. Pagelets), die inhaltlich in sich geschlossene Informationseinheiten darstellen sollen. Tritt eine Blockstrukturen in den Dokumenten häufiger auf, so handelt es sich um templategenerierten, also zusätzlichen Inhalt. Auch der InfoDiscoverer Algorithmus [LH02] und der SST Ansatz [YLL03] basieren auf der Idee häufig wiederkehrende Elemente in der Dokumentsammlung zu erkennen und verwenden lediglich andere Methoden. Dadurch, dass im TD zusätzliches Wissen aus mehreren Dokumente bezogen wird, erzielen diese Ansätze in der Regel bessere Ergebnisse als CE Verfahren die nur ein einzelnes Dokument betrachten.

Der Erfolg der TD Algorithmen hängt allerdings stark von der Qualität der Trainingsmenge ab, also der Dokumentsammlung aus der eine gemeinsame Templatestruktur abgeleitet wird. Enthält die Trainingsmenge Dokumente die auf verschiedenen Templates basieren, so kann es passieren, dass die TD Algorithmen keine, unvollständige oder gar fehlerhafte Templatestrukturen ableiten. Daher werden die Trainingsdatensätze häufig von Hand erzeugt oder zumindest von einem Menschen überprüft. Diese manuellen Eingriffe sind bei CE Verfahren nicht nötig, da sie komplett ohne Trainingsmenge auskommen.

Will man also die Vorteile der beiden Welten miteinander verbinden, so benötigt man letztendlich einen Algorithmus der automatisch qualitativ hochwertige TD Trainingsmengen erzeugen kann. Geht man – wie beim CE – von einem einzelnen initialen Startdokument aus, so kann man diesen Prozess in zwei Schritte aufteilen: das Sammeln von potentiell geeigneten Dokumenten und das Herausfiltern jener Dokumente die auf einem anderen Template basieren. Der erste Teil, das Sammeln geeigneter Dokumente, kann recht einfach durch einen lokalen Crawler realisiert werden, der alle im initialen Dokument referenzierten Dokument aus dem Web läd[1].

Das Herausfiltern von Dokumenten die ein anderes Template verwenden, ist etwas schwieriger. Dazu müsste man eigentlich die Template Strukturen kennen; diese werden jedoch erst durch das nachgelagerte TD erkannt. Der Ausweg aus diesem Henne-Ei Dilemma besteht darin, die Dokumente auf strukturelle Ähnlichkeit hin zu untersuchen, da Templates sehr stark die Struktur eines Dokumentes beeinflussen.

Ein klassischer Ansatz zur Bestimmung von strukturellen Ähnlichkeiten in Webdokumenten setzt auf dem DOM-Baum auf. Dabei werden werden Tree Edit Distances verwendet, wie beispielsweise im RTDM Algorithmus [RGdL04]. Aber selbst dieser optimierte Ansatz hat schlimmstenfalls quadratischen Aufwand. Andere Ähnlichkeitsmaße lassen sich schneller berechnen, erfassen dabei jedoch nur noch Teilaspekte der Baumstruktur.

Im Vergleich verschiedener Ähnlichkeitsmaße für DOM Strukturen und möglicher darauf angewendeter Cluster-Verfahren stellte sich erstaunlicherweise heraus, dass gerade einige der schneller zu berechnenden Maße hervorragende Ergebnisse liefern, sogar bessere als RTDM [Got08b]. Am besten schnitt dabei ein Ansatz ab, der in diesem Zusammenhang speziell neu auf die Templateerkennung übertragen wurde. Dazu wird die Abfolge der HTML Tags im Quellcode als eine lange Sequenz betrachtet. Diese Sequenz wird in Teilstücke fester Länge zerlegt – sogenannte Shingles [BGMZ97]. Die Struktur zweier Dokumente wird dann über die Mächtigkeit der Schnittmenge der gemeinsamen Tagsequenz Shingles im Verhältnis zur Größe der Shinglemengen der einzelnen Dokumente verglichen. Je größer die Übereinstimmung in der Struktur desto größer ist dieses Verhältnis. In Abbildung 3 sieht man gut, wie in einer Testkollektion von 500 HTML Dokumenten die auf fünf verschiedene Templates basieren die strukturell ähnliche Dokumente unter RTDM und Tag Shingle Abstand zusammengruppiert werden. In den Untersuchungen ergab sich weiterhin, dass sich Single-Link Cluster Algorithmen bestens eigenen, um diese Gruppen sicher ausfindig zu machen. Auf den Testdaten von 500 Dokumenten lieferte die Kombination aus Single-Link Clusterung und dem Tag Shingle Ähnlichkeitsmaße eine perfekte

[1] Templates werden wie eingangs erwähnt meistens in Content Management Systemen eingesetzt. In diesen wiederum sind die Dokumente sehr stark untereinander mit Hyperlinks verknüpft, weshalb die Chancen gut stehen, dass die verlinkten Dokumente zu einem großen Teil das gleiche Template verwenden.

Abbildung 3: Gruppierung von 500 Dokumenten mit fünf verschiedenen Template bei Anwendung des Tag Shingle und des RTDM Ähnlichkeitsmaßes

Gruppierung der Dokumente, d.h. es wurde kein einziger Fehler gemacht.

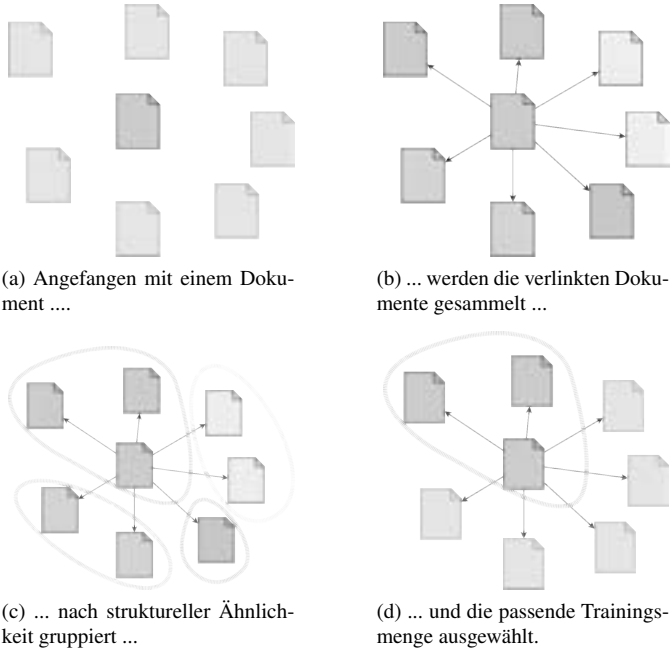
Damit lässt sich nun jedes TD Verfahren wie ein CE Verfahren behandelt, das auf einem isolierten Dokument arbeitet [Got08a]. Das Vorgehen wird in Abbildung 4 erläutert. Vom initialen Dokument ausgehend werden zunächst alle verlinkten Dokumente bestimmt. Eine Template Cluster Analyse ergibt die Gruppen der Dokumente, die auf der gleichen Vorlage beruhen. Wählt man nun die Gruppe aus, in der das initiale Dokument liegt, so hat man eine qualitativ hochwertige Trainingsmenge für genau das Dokument, auf dem man Template Detection durchführen wollte. Der gesamte Prozess läuft vollautomatisch und ohne das Zutun eines Nutzers ab.

5 Zusammenfassung und Ausblick

Die Bestimmung des Hauptinhaltes in Webdokumenten ist in vielen Anwendungsbereichen ein Anliegen. Die geleisteten Beiträge in diesem Bereich umfassen Evaluationsmethoden, neue CE Algorithmen die bezüglich ihrer Performanz Maßstäbe setzen, sowie ein Ansatz zur Template Clusterung der vollautomatisch qualitativ hochwertige Trainingsmengen für TD Ansätze erzeugen kann und damit die Brücke zwischen den CE Verfahren auf einzelnen Dokumenten und Trainingsmengenbasierten TD Algorithmen schlägt.

Diese Ergebnisse haben bereits Anwendung in der Praxis gefunden. Sowohl in eigenen als auch in Arbeiten anderen Gruppen wurde ACCB zur Inhaltsextraktion eingesetzt, um beispielsweise Text Mining zu unterstützen, die Lesbarkeit von HTML Dokumenten zu bestimmen oder Nutzer bei der Relevanzeinschätzung von Webdokumenten zu unterstützen. Mit Hilfe der Ähnlichkeitsmaße für Templatestrukturen konnten Verfahren entwickelt werden um Redesigns auf Webseiten zu erkennen.

Neben diesen Anwendungen wird auch an den Algorithmen und Verfahren selbst weiterentwickelt. So konnten erfolgreich evolutionäre Ansätze zur Parameteroptimierung ein-



(a) Angefangen mit einem Dokument ....

(b) ... werden die verlinkten Dokumente gesammelt ...

(c) ... nach struktureller Ähnlichkeit gruppiert ...

(d) ... und die passende Trainingsmenge ausgewählt.

Abbildung 4: Automatisches Erstellen einer Trainingsmenge für Template Detection.

gesetzt und verschiedene CE Verfahren zur Verbesserung der Gesamtleitsung kombiniert werden. Diese Themen sind Gegenstand aktuelle Forschungsbestrebungen und erste Ergebnisse werden in Kürze vorgestellt werden.

Literatur

[BGMZ97] Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse und Geoffrey Zweig. Syntactic Clustering of the Web. *Computer Networks*, 29(8-13):1157–1166, 1997.

[BYR02] Ziv Bar-Yossef und Sridhar Rajagopalan. Template detection via data mining and its applications. In *Proceedings of the 11th International Conference on World Wide Web*, Seiten 580–591, New York, NY, USA, 2002. ACM Press.

[CMZ03] Yu Chen, Wei-Ying Ma und Hong-Jiang Zhang. Detecting web page structure for adaptive viewing on small form factor devices. In *Proceedings of the 12th international conference on World Wide Web*, Seiten 225–233, New York, NY, USA, 2003. ACM Press.

[FKS01] Aidan Finn, Nicholas Kushmerick und Barry Smyth. Fact or Fiction: Content Classification for Digital Libraries. In *DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries*, 2001.

[GKNG03] Suhit Gupta, Gail Kaiser, David Neistadt und Peter Grimm. DOM-based Content Extraction of HTML documents. In *Proceedings of the 12th International Conference on World Wide Web*, Seiten 207–214, New York, NY, USA, 2003. ACM Press.

[Got07] Thomas Gottron. Evaluating Content Extraction on HTML Documents. In *Proceedings of the 2nd International Conference on Internet Technologies and Applications*, Seiten 123–132, 2007.

[Got08a] Thomas Gottron. Bridging the Gap: From Multi Document Template Detection to Single Document Content Extraction. In *Proceedings of the EuroIMSA Conference on Internet and Multimedia Systems and Applications 2008*, Seiten 66–71. ACTA Press, Calgary, 2008.

[Got08b] Thomas Gottron. Clustering Template Based Web Documents. In *ECIR '08: Proceedings of the 30th European Conference on Information Retrieval*, Seiten 40–51. Springer, 2008.

[Got08c] Thomas Gottron. Content Code Blurring: A New Approach to Content Extraction. In *DEXA '08: 19th International Workshop on Database and Expert Systems Applications*, Seiten 29 – 33. IEEE Computer Society, 2008.

[Got08d] Thomas Gottron. *Content Extraction: Identifying the Main Content in HTML Documents*. Dissertation, Johannes Gutenberg-University, Mainz, 2008.

[LH02] Shian-Hua Lin und Jan-Ming Ho. Discovering Informative Content Blocks from Web Documents. In *Proceedings of the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Seiten 588–593, New York, NY, USA, 2002. ACM Press.

[MOC05] Constantine Mantratzis, Mehmet Orgun und Steve Cassidy. Separating XHTML content from navigation clutter using DOM-structure block analysis. In *Proceedings of the sixteenth ACM conference on Hypertext and hypermedia*, Seiten 145–147, New York, NY, USA, 2005. ACM Press.

[PBC+02] David Pinto, Michael Branstein, Ryan Coleman, W. Bruce Croft, Matthew King, Wei Li und Xing Wei. QuASM: a system for question answering using semi-structured data. In *Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, Seiten 46–55, New York, NY, USA, 2002. ACM Press.

[RGdL04] D. C. Reis, P. B. Golgher, A. S. da Silva und A. F. Laender. Automatic Web News Extraction using Tree Edit Distance. In *Proceedings of the 13th International Conference on World Wide Web*, Seiten 502–511, New York, NY, USA, 2004. ACM Press.

[VR79] C. J. Van Rijsbergen. *Information Retrieval*. Butterworths, 2nd. Auflage, 1979.

[YLL03] Lan Yi, Bing Liu und Xiaoli Li. Eliminating Noisy Information in Web Pages for Data Mining. In *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Seiten 296–305, New York, NY, USA, 2003. ACM Press.

Thomas Gottron, studierte Mathematik, Betriebswirtschaftslehre und Informatik an der Johannes Gutenberg-Universität in Mainz sowie an der Glasgow University in Schottland. Nach seiner Diplomarbeit über Web Content Management Systemen beschäftigte er sich während der Promotion im Bereich der Informatik hauptsächlich mit den Themen Content Extraction, Template Clustering und Inhaltsanalyse in Webdokumenten.