# Webscraping als Methode der Informationsbeschaffung

*Jan Seipel*

Für Journalisten bietet das WWW im Vergleich zu früher ungeahnte Möglichkeiten der Recherche und Themenfindung. Noch nie waren so viele Informationen öffentlich einsehbar wie heute. Auf der anderen Seite sind die gewünschten Inhalte an vielen unterschiedlichen Orten im Web verteilt, auf jede mögliche oder unmögliche Weise formatiert oder aufgrund schlechter Informationsarchitektur nur mit größerem Aufwand zu erreichen. Je nach Thema kann die reine Fülle an Rechercheergebnissen schlichtweg überfordern und nicht immer sind alle relevanten Inhalte über gängige Suchmaschinen auffindbar. Das bedeutet dann oft mühsam in nicht immer nutzerfreundlich gestalteten Datenbanken nach den relevanten Informationen zu wühlen.

Was für Journalisten gilt, gilt umso mehr für Informationsspezialisten, sofern sie sich – wie im Fall von Mediendokumentaren – als deren Dienstleister verstehen. Ging es vor Jahren noch darum, einem prinzipiellen Mangel an Informationen durch das Vorhalten entsprechender Ressourcen zu begegnen, besteht die Aufgabe heute vielfach darin, die sprichwörtliche „Nadel im Heuhaufen“ zu finden. Meist geht das nicht ohne einen gewissen Grad an Automatisierung. Viele Dokumentationsabteilungen verfügen deshalb inzwischen über eine Reihe von Werkzeugen, die ungeordnete Informationsströme im Web kanalisieren, filtern und in einem nutzerfreundlicheren Format ausgeben können.

Technisch bedient man sich hierfür häufig vorhandener Schnittstellen, die die jeweiligen Quellen zur Verfügung stellen. Neben anwendungsspezifischen APIs sind dies zum Beispiel RSS- oder Atom-Feeds, die in eigene Datenbanken geladen und von dort aus weiterverarbeitet werden können. Die Informationen liegen in diesen Fällen also bereits in einem strukturierten Format vor und enthalten wenig bis gar keinen überflüssigen Content. Außerdem ist ihre Anwendung vergleichsweise einfach, sodass man häufig auch ohne Programmierkenntnisse zu zufriedenstellenden Ergebnissen kommt.

So komfortabel dieser Ansatz auch ist: Wenn keine der genannten Schnittstellen angeboten werden, bleibt die ständige Beobachtung dieser Inhalte schwierig. Sofern es sich bei den Quellen um einfach strukturierte HTML-Seiten handelt, kann man sich zwar gelegentlich mit verschiedenen Tools weiterhelfen. Gerade bei öffentlichen Datenbanken, deren Inhalte über Webformulare abgefragt werden („Deep Web“), ist dies jedoch meistens nicht möglich. Hier können Webscraper weiterhelfen, die das automatisierte „Auslesen“ von bestimmten strukturierten oder semi-strukturierten Inhalten aus öffentlich zugänglichen Webseiten ermöglichen.

Jan Seipel
IID Potsdam / SWR
janfseipel@
gmail.com

## WIE FUNKTIONIEREN WEBSCRAPER?

Wörtlich übersetzt bedeutet Webscraper „Netzkratzer“ – und das beschreibt das, was diese Programme tun, eigentlich schon ganz treffend. Sie laden eigenständig Webseiten in den Arbeitsspeicher, suchen dort nach bestimmten Mustern und Informationen und „kratzen“ diese aus dem Quelltext heraus. Damit ähnelt die Funktionsweise der von Webcrawlern für Suchmaschinen. Im Unterschied zu diesen besucht ein Scraper aber meist nur eine begrenzte Anzahl von URLs und sein Ziel ist es nicht, die Relevanz einer Webseite zu beurteilen, sondern nach bestimmten Informationen zu suchen und diese in ein strukturiertes Format zu übersetzen.

Die meisten Scraper nutzen für die Lokalisierung von Informationen die hierarchische Struktur von HTML-Dokumenten. Über standardisierte Abfragesprachen (z.B. XPath, CSS-Selektoren, Reguläre Ausdrücke) kann das Skript auf einzelne Elemente zugreifen und diese weiterverarbeiten. Um Informationen effizient aus HTML-Dokumenten auslesen zu können, ist das Erlernen einen Skript-Sprache empfehlenswert. Zwar existiert eine Reihe von Tools, die versprechen, das Scrapen auch ohne Programmierkenntnisse zu ermöglichen. Allerdings erwiesen sich die meisten von ihnen in der Praxis als zu

*Vortragsmanuskript (gehalten auf der Frühjahrstagung des vfm am 26. April 2016)

| *Anforderung* | **Beschreibung** |
|---|---|
| ***Navigation in HTML*** | Um die relevanten Tags eindeutig zu identifizieren, muss ein Programm die hierarchische Struktur von HTML-Dokumenten verarbeiten können. |
| ***Reguläre Ausdrücke*** | Sofern das Dokument nicht ausreichend durch Tags strukturiert ist, muss ein Scraper auf NLP-Methoden zurückgreifen können. Zumindest der Umgang mit Regulären Ausdrücken sollten implementiert sein. |
| ***Rekursives Scrapen*** | Scraper müssen fähig sein, relevante Links auf einer Seite zu erkennen und ihnen automatisch zu folgen. |
| ***JavaScript-Fähigkeit*** | Beim Zugriff auf eine Webseite müssen automatisch clientseitige Skriptsprachen ausgeführt werden. |
| ***Interaktionsfähigkeit*** | Vor allem für Deep-Web-Anfragen muss ein Scraper in der Lage sein, beispielsweise automatisiert Formularanfragen abzuschicken. |
| ***Relationen speichern*** | Um aus einer Sammlung von Webseiten strukturierte Informationen zu generieren, müssen die Beziehungen der Daten zueinander erhalten bleiben. |
| ***Daten säubern*** | Um unnötigen Arbeitsaufwand zu vermeiden, sollten Scraper bereits während des Extraktionsvorgangs in der Lage sein, unsaubere Formatierungen zu erkennen und zu beseitigen. |
| ***Dopplungen vermeiden*** | Es sollte sichergestellt sein, dass Informationen, die bereits ausgelesen wurden, in Zukunft nicht erneut eingesammelt werden. |
| ***Daten ausgeben*** | Um mit den gewonnenen Informationen arbeiten zu können, müssen Scraper eine sinnvolle Möglichkeit der Speicherung und Datenausgabe bereitstellen. |
| ***Exception-Handling*** | Ein Scraper sollte mit unvorhergesehenen Ereignissen (z. B. nicht erreichbaren Webseiten) umgehen können, ohne dass der Scrapingvorgang insgesamt abbricht. |

Tabelle 1: Anforderungen an einen Webscraper

unflexibel, um mit der Vielzahl an möglichen Seitenstrukturen umzugehen. Im Zuge der Programmierung sollte ein leistungsfähiger Scraper folgenden Anforderungen erfüllen: (Vgl. Tabelle)

Aus den zusammengestellten Anforderungen lässt sich eine abstrakte Programmstruktur ableiten. Diese ist in Abbildung 1 in Form eines Programmablaufplans dargestellt. In dem Modell wird von dem einfachen Fall ausgegangen, dass alle Links zu Seiten, die relevante Inhalte bereitstellen könnten, auf einer einzelnen „Startseite“ hinterlegt sind. Eine solche kann beispielsweise in Form einer Trefferliste als Antwort einer Suchanfrage vorliegen.

Nach dem Einlesen der Seite extrahiert das dargestellte Skript alle als relevant eingestuften URLs und prüft für jede einzelne, ob die entsprechende Seite bereits besucht wurde. Hierfür kommt ein Datensatz zum Einsatz, in dem alle bereits besuchten Links hinterlegt sind. Ist die geprüfte URL bereits vorhanden, wird sie ignoriert und der nächste Link einer Prüfung unterzogen. Sofern es sich um einen neuen Link handelt, wird die entsprechende Seite geladen. Finden sich hier relevante Inhalte, werden diese extrahiert, aufbereitet und temporär in einem Format gespeichert, das gewährleistet, dass die auf der Seite hinterlegten Relationen zwischen den extrahierten Daten vorhanden bleiben. Die entsprechende URL wird dem Datensatz mit den bereits besuchten Links hinzugefügt. Sofern keine Daten vorhanden sind, wird die URL ebenfalls gelöscht und die Schleife beginnt von vorne.

Die extrahierten Daten werden dann in einer dem Zweck angemessenen Form gespeichert – beispielsweise in einer relationalen Datenbank – und stehen für die weitere Nutzung zur Verfügung. Die beschriebenen Vorgänge werden in verschiedenen Schleifen so lange wiederholt, bis sämtliche anfangs extrahierten URLs entweder gelöscht oder besucht wurden. Sobald dies der Fall ist, wird der Prozess beendet und der gewonnene Datensatz kann weiterverarbeitet werden.

## WOZU LASSEN SICH WEBSCRAPER NUTZEN?

Der Einsatz von Webscrapern kann immer dann sinnvoll sein, wenn 1.) eine große Menge an gleichförmig strukturierter Information auf vielen verschiedenen Webseiten verteilt vorliegt oder 2.) in regelmäßigen Abständen neue Informationen auf einer Webseite veröffentlicht werden und diese keine Schnittstelle für den automatisierten Abruf bereit stellt. Ein regelmäßiges Monitoring kann hierdurch erleichtert oder sogar erst ermöglicht werden, da entsprechende Quellen nicht mehr einzeln besucht werden müssen.

Jedoch ist die Generierung strukturierter Datensätze aus dem Inhalt von Webseiten nicht nur für Monitoring- und Recherchezwecke interessant. Der in den letzten Jahren immer stärker in den Fokus gerückte Datenjournalismus hat den Bedarf an maschinenlesbaren Informationen innerhalb von Medienunternehmen deutlich gemacht. Gerade für die Suche nach neuen Themen bietet es sich an, nicht nur auf Quellen

zuzugreifen, die von öffentlicher Seite bereits in entsprechenden Formaten zur Verfügung gestellt werden. Interessante neue „Stories“ verbergen sich oftmals gerade in schwer zugänglichen Datensammlungen, die erst in die entsprechende Form gebracht werden müssen. Webscraping kann solche Datenbestände mit überschaubarem Aufwand zugänglich machen.

## FALLBEISPIEL: EIN SCRAPER FÜR DEN LANDTAG BW

Für eine Abschlussarbeit am Institut für Information und Dokumentation an der Fachhochschule Potsdam (IID), die ein zweijähriges Volontariat beim Südwestrundfunk beendete, sollte der praktische Einsatz eines Webscrapers getestet werden. Es stellte sich die Frage, ob durch den Einsatz eines solchen Skripts der Zugriff auf die Dokumentation der Parlamentsvorgänge des Landtags Baden-Württemberg für die Sendeanstalt vereinfacht werden könnte. Für diese Aufgabenstellung waren mehrere Gründe ausschlaggebend: Für das Sendegebiet des Südwestrundfunks hat der baden-württembergische Landtag prinzipiell Relevanz. Während die Vorgänge im Bundestag von vielen großen Medien beobachtet werden, gilt dies für die Landesparlamente nicht im gleichen Maße.

Der Zugriff auf die Datenbank erfolgt in diesem Fall über sogenannte „Query Strings“. Dabei wird die URL um verschiedene Parameter erweitert, auf deren Grundlage dann die gewünschten Daten ausgegeben werden. Dies vereinfachte die Programmierung des Scrapers.

Die Metadaten zu den im Dokumentationssystem verzeichneten Vorgängen enthalten neben Formalangaben auch die vergebenen Schlagworte. Diese könnten zu einem späteren Zeitpunkt für Filterprozesse zur Verbesserung der Precision genutzt werden.

Realisiert wurde der Scraper mit Hilfe der Programmiersprache Python. Diese Skriptsprache besitzt eine vergleichsweise einfache Syntax und ist bekannt für ihre gute Lesbarkeit. Die Testläufe für das Skript erfolgten von einem Server der Hauptabteilung Information, Dokumentation und Archive (IDA) des Südwestrundfunks aus, wobei auf eine virtuelle Browserlösung zurückgegriffen wurde. Damit konnte auch der Quelltext dynamischer Seiten fehlerfrei geladen werden. Die ausgelesenen Daten wurden schlussendlich in eine relationale Datenbank überführt.

Insgesamt brachte der Programm-Prototyp gute Ergebnisse. Fehlermeldungen und damit verbundene Programmabbrüche blieben aus, Qualität und Vollständigkeit der ausgelesenen Daten schienen nach einer stichprobenartigen Kontrolle gewährleistet zu sein. Um dies abschließend beurteilen zu können, wäre allerdings ein automatisierter Testlauf von mehreren Wochen mit anschließendem systematischem Abgleich der Ergebnisse mit den Originaldaten notwendig gewesen. Ein Solcher konnte in der Kürze der Zeit jedoch nicht durchgeführt werden. Auch die

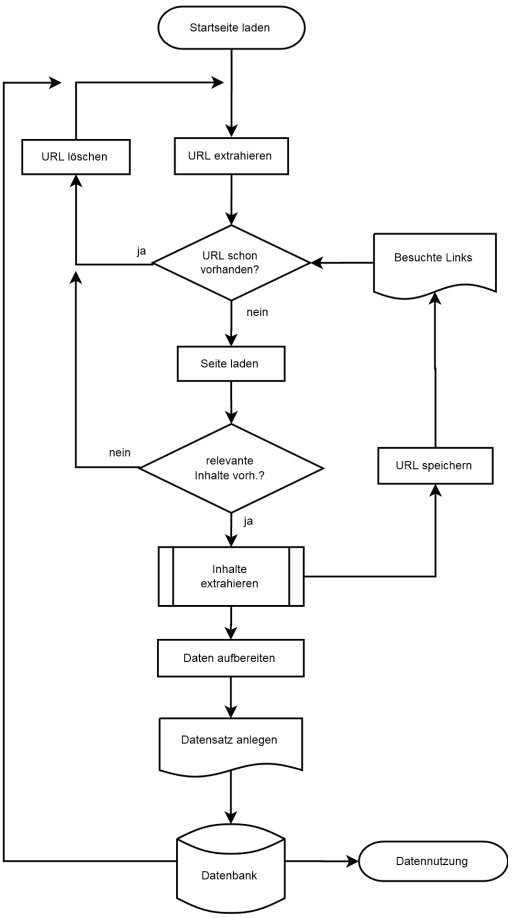


Diagramm: Funktionsweise eines einfachen Webscrapers

ursprüngliche Überlegung, die Fachredaktionen des SWR mit Hilfe eines Push-Dienstes über die Neuveröffentlichung relevanter Parlamentsvorgänge pro-aktiv zu informieren (z.B. über E-Mail-Alerts), wurde bis zum Erscheinen dieses Artikels nicht umgesetzt.

## FAZIT

Die Notwendigkeit, die Fülle an Informationen zu bewältigen, die das WWW bereithält und die mit Sicherheit in Zukunft weiter wachsen wird, geht auch an Informationsspezialisten nicht vorbei. Um sie zu bewältigen, muss künftig wohl immer häufiger auf technische Hilfsmittel zurückgegriffen werden, die automatisiert Filter- und Ordnungsfunktionen übernehmen bzw. einen sinnvoll verwendbaren Zugang zu Informationsquellen erst ermöglichen. Webscraper können eines dieser Hilfsmittel sein. Für Dokumentationsabteilungen lohnt es sich sicherlich, entsprechende Kompetenzen zu entwickeln. Noch wichtiger erscheint allerdings ein fundiertes Wissen über die technischen Voraussetzung moderner Informationsvermittlung, um diese Hilfsmittel selbst entwerfen und flexibel an die jeweiligen Informationsbedürfnisse anpassen zu können. Eine zumindest grundlegende Programmierkompetenz wäre dann notwendig. In Zeiten, in denen regelmäßig von Journalisten gefordert wird, selbst „zu coden“, sollte dies auch für Mediendokumentare als deren Dienstleister gelten. •