Weg vom manuellen Speichern: RSS-Feeds in der automatisierten Datenerhebung bei Online-Medien

Trilling, D.

Publication date
2014
Document Version
Author accepted manuscript
Published in
Automatisierung in der Inhaltsanalyse

Weg vom manuellen Speichern: RSS-Feeds in der automatisierten Datenerhebung bei Onlinemedien

Damian Trilling

Immer häufiger haben kommunikationswissenschaftliche Inhaltsanalysen nicht Printmedien, sondern Onlinemedien zum Untersuchungsgegenstand. Dies bietet zum einen neue Möglichkeiten, wirft aber auch Fragen auf. Vor allem das Bestimmen der Grundgesamtheit und das Ziehen von Stichproben gestalten sich komplexer als bei Printmedien. Ganz unabhängig von der beinahe trivialen Feststellung, dass es unmöglich ist, die Berichterstattung „im Internet" in seiner Gänze zu erfassen, stellt sich auch bei einer zuvor festgelegten Auswahl an Websites, deren Berichterstattung analysiert werden soll, die Frage, wie die Gesamtheit der erschienen Texte ermittelt werden kann.
Im nächsten Schritt muss entschieden werden, wie das Analysematerial erfasst, gespeichert und für den Codierprozess zur Verfügung gestellt werden kann.
Bei der klassischen Inhaltsanalyse von Printmedien ist der Untersuchungsgegenstand statisch und kann in der Regel in Archiven jederzeit abgerufen werden. Von wenigen Ausnahmen abgesehen, in denen beispielsweise nach Andruck noch Artikel ausgetauscht werden oder sich Regionalausgaben auch im überregionalen Teil unterscheiden, besteht kein Zweifel über die Grundgesamtheit der Artikel einer Ausgabe. Eine Stichprobe kann einfach und zuverlässig gezogen werden: Schließlich ist die Anzahl der Ausgaben pro Zeitraum bekannt und jeder Artikel kann eindeutig exakt einer Ausgabe zugeordnet werden. Hierzu existieren zudem eine umfangreiche Methodenliteratur und entsprechende Lehrbücher (z.B. Früh, 2007; Krippendorf, 2004; Merten, 1995; Rössler, 2005).
Online-Nachrichtenseiten werden im Gegensatz dazu kontinuierlich upgedatet. Was morgens der Aufmacher war, ist abends möglicherweise nur noch eines von vielen Themen oder gar nicht mehr online. Ebenso kann ein Artikel mehrere Tage auf der Homepage bleiben.
Wie aber können unter diesen Rahmenbedingungen effizient und gleichzeitig reliabel Daten erhoben werden? Der vorliegende Beitrag widmet sich zunächst theoretischen Vorüberlegungen zur inhaltsanalytischen Erfassung von Online-Nachrichtenmedien und stellt im zweiten Teil eine praktische Methode vor, die es erlaubt, mit geringem Aufwand die Daten für eine solche Analyse zu erheben.

1. Unterschiede in der Inhaltsanalyse von Online-Nachrichten und Printmedien

Bereits vor etwas mehr als einem Jahrzehnt haben Rössler und Wirth (2001) Herausforderungen, die sich bei der Inhaltsanalyse von Webangeboten ergeben, zusammengestellt. Sie nennen unter

anderem die Archivierbarkeit, die schwierige Codierung multimedialer Inhalte und die Frage, wie dynamische Inhalte – also Inhalte, die erst durch den Aufruf des Benutzers erzeugt werden und automatisch auf diesen zugeschnitten werden – inhaltsanalytisch erfasst werden können (vgl. Rössler & Wirth, 2001). Je nach Untersuchungsgegenstand und Fragestellung spielen diese und andere Herausforderungen jedoch eine unterschiedlich große Rolle: Nicht jeder Untersuchungsgegenstand beinhaltet multimediale oder dynamische Inhalte. Daher greift der vorliegende Beitrag einen Typ der Online-Inhaltsanalyse heraus, um vertiefend auf dessen Spezifika einzugehen: die Analyse von textbasierten Nachrichtenangeboten. Nachrichtenangebote werden dabei sehr breit definiert und verstanden als Websites, auf denen regelmäßig aktuelle Texte veröffentlicht werden, sei es auf Zeitungswebsites, Online-Only-Nachrichtensites, Nachrichtensuchmaschinen oder Blogs. Auch wenn einige Überlegungen auch auf andere Bereiche zutreffen, so wird doch vor allem von Analysen im Bereich der politischen Kommunikationsforschung und der Journalismusforschung ausgegangen. Im Gegensatz zu dezidiert Web-spezifischen Studien wie beispielsweise Netzwerkanalysen von Linkstrukturen handelt sich also um „klassische“ Forschungsdesigns (Schweitzer, 2010, p. 61), die die Analyse von Texten zum Ziel haben, die mit einer gewissen Periodizität erscheinen und erneuert werden, wie eben journalistische Webangebote oder Blogs.

Solche Analysen weisen in Zielsetzung und Operationalisierung starke Parallelen mit herkömmlichen Inhaltsanalysen von Printmedien auf, und einmal erfasst, können die Texte auf ähnliche Weise analysiert werden. Im Folgenden wird diskutiert, was bei der Erfassung beachtet werden muss, welche Hürden im Vergleich zur klassischen Print-Inhaltsanalyse zu überwinden sind und welche zusätzlichen Möglichkeiten sich bieten. Der Beitrag fokussiert daher auf Bausteine für die Konzeption und Durchführung einer Inhaltsanalyse von textbasierten Nachrichtenangeboten im Internet und zeigt Vor- und Nachteile verschiedener Herangehensweisen auf[1].

1.1 Kontinuierliche Updates

Online-Nachrichtenmedien haben keinen Redaktionsschluss, sondern werden kontinuierlich aktualisiert. Für die Inhaltsanalyse eines solchen „moving target“ (McMillan, 2000) gilt, dass die Untersuchungseinheit nicht so einfach bestimmt werden kann wie bei Analysen von Printmedien

[1] Fragestellungen, die sich nicht auf den Nachrichtentext richten, stehen nicht im Mittelpunkt dieses Beitrags. Dazu zählen z.B. die exakte grafische Wiedergabe, dessen Erfassung durch die immer größer werdende Vielfalt an Endgeräten mit großen Schwierigkeiten einhergeht – was nicht nur ein unerwünschter technischer Nebeneffekt sein kann, sondern von vielen Anbietern aktiv im Rahmen des sogenannten *Responsive Design*, dem automatischen Anpassen einer Website an das Endgerät, herbeigeführt wird (vgl. z.B. Marcotte, 2010). Auch die Analyse der Struktur und Gestaltung einer Website in ihrer Gesamtheit (vgl. z.B. Schweitzer, 2010) soll hier nicht weiter interessieren. Gleiches gilt für inhaltsanalytische Probleme, die sich aus der individuellen und nicht intersubjektiv replizierbaren Generierung von Anzeigen ergeben.

(vgl. z.B. Zeller & Wolling, 2010). Da also das Konzept der statischen Ausgabe, die in diskreten Zeitabständen durch eine neue Ausgabe ersetzt wird, nicht auf Online-Nachrichtenseiten übertragen werden kann, haben sich kommunikationswissenschaftliche Studien häufig mit Notlösungen beholfen: Beispielsweise wird die zu untersuchende Website täglich um eine bestimmte Uhrzeit manuell oder teilweise automatisiert gespeichert. Neben dem immensen Aufwand sind vor allem die Arbitrarität der Wahl des Zeitpunktes und damit einhergehend das Risiko, kurzlebige Artikel zu verpassen, aus methodologischer Sicht problematisch.

Auf regelmäßig aktualisierten Websites werden Artikel werden heutzutage in aller Regel nicht mehr manuell eingestellt. Stattdessen nutzt man sogenannte Content Management Systems, bei denen der Schreiber über eine Benutzeroberfläche den Text in eine Datenbank eingibt, aus der dann wiederum die Website gespeist wird. Um das Problem der kontinuierlichen Updates zu umgehen, haben einige Forscher daher als elegante Lösung diese Datenbanken direkt ausgelesen – also eine Kopie der Daten angefertigt, aus denen die Website-Inhalte gespeist werden (z.B. Sjøvaag & Stavelin, 2012). Das setzt jedoch die Kooperation der Betreiber der untersuchten Site voraus und dürfte zudem praktisch kaum durchführbar sein, sobald mehrere Websites mit völlig unterschiedlicher technischer Infrastruktur analysiert werden müssen. Das Vorgehen ist im Grunde das moderne Äquivalent zu den ersten teilautomatischen Inhaltsanalysen der 60er und 70er Jahre, für die man sich Zugang zu Lochkarten oder Dateien von Setzereien verschafft hatte (Bröker, 1981).

Andere Forscher haben versucht, den kontinuierlichen Updates Rechnung zu tragen, indem sie nicht einmal, sondern zweimal (z.B. Quandt, 2008) oder sogar dreimal täglich den Untersuchungsgegenstand erfassten (z.B. Lim, 2010) – was neben dem immensen Aufwand auch in eine riesige Datenmenge zur Folge hat, die zudem noch von Duplikaten befreit werden muss.

1.2 Rückwirkendes Erfassen

Während es bei der Analyse von Printmedien prinzipiell egal ist, ob die aktuellsten oder weit zurückliegende Ausgaben codiert werden sollen, stellt die rückwirkende Erhebung von Online-Nachrichtenmedien noch immer eine große Schwierigkeit dar: Während durch Bibliotheken, Pressearchive und entsprechende Verlagsangebote in der Regel eine weitreichende Zugänglichkeit zum Untersuchungsmaterial gewährleistet ist, besteht noch keine vergleichbare Infrastruktur für Online-Medien, zumindest nicht in diesem Umfang. Anekdotisches Material lässt sich über die Wayback Machine (http://archive.org) gewinnen, und einige professionelle Nachrichtenmedien bieten eigene Archive an. Eine flächendeckende Infrastruktur, in der verschiedene Websites mit einheitlichen Standards archiviert würden, existiert jedoch nicht.

1.3 Metadaten

Neben den inhaltlichen Variablen des Codebuchs, denen das eigentliche Forschungsinteresse gilt, müssen in aller Regel auch einige Metadaten erfasst werden, schon allein, um Dokumentationsanforderungen zu genügen. Das können einfach festzustellende Parameter wie die URL sein, aber auch Daten wie das Publikationsdatum, das sich bei einem manuellen Vorgehen nicht erfassen lässt, sofern es nicht im Text des Artikels angeben ist. Doch selbst wenn solche Daten verfügbar sind, ist das manuelle Erfassen dieser Daten fehleranfällig und aufwändig. Wünschenswert ist daher, Metadaten wie den Publikationszeitpunkt automatisch an jeden erfassten Beitrag zu koppeln.

1.4 Speichern/Sicherung

Auch die Speicherung und Archivierung des Materials erweist sich als komplexer als bei Printprodukten (z.B. Meier, Wünsch, Pentzold, & Welker, 2010; Schweitzer, 2010). Es muss beispielsweise festgelegt werden, in welcher Form die Speicherung erfolgen soll. Frühe Studien arbeiteten mit Papierausdrucken, was nicht nur mit ökonomischen und ökologischen Kosten verbunden ist, sondern auch die vielfältigen Möglichkeiten einer späteren (teil-)automatisierten Analyse unmöglich macht. Daneben lassen sich interaktive Elemente nicht codieren, Information geht verloren. Mit ähnlichen Nachteilen behaftet ist auch das Anfertigen von Screenshots, das ebenfalls praktiziert wurde. Heute werden zunehmend teilautomatisierte Prozesse verwendet und programmierbare Crawler eingesetzt (z.B. Quandt, 2008; Rüdiger & Welker, 2010). Ein solches Programm ist in der Lage, eine Liste von Internetseiten herunterzuladen und dabei auch – entsprechend der Vorgaben der Benutzer – angetroffenen Links zu folgen und diese Seiten ebenfalls herunterzuladen. Größter Vorteil ist die Effizienz und die Möglichkeit, sehr große Seitenanzahlen herunterzuladen. Schwierigkeiten kann es jedoch bereiten, den Crawler so zu programmieren, dass zwar alle benötigten Inhalte, aber möglichst wenig irrelevante Inhalte heruntergeladen werden. Zudem ist es notwendig, den für die Inhaltsanalyse interessanten Inhalt von beispielsweise Navigationselementen oder Werbung zu trennen (siehe hierzu auch den Beitrag von Günther & Scharkow, 2012)

1.5 Ein Anforderungskatalog

Aus den hier geschilderten Charakteristika bei der Erhebung von inhaltsanalytisch zu untersuchenden Daten von Online-Medien lässt sich ein Anforderungskatalog für ein Online-Datenerhebungsverfahren ableiten. Wünschenswert sind:

1. eine automatische Datenerhebung mit so wenig manuellen Schritten wie möglich, um auch die Erfassung großer Fallzahlen möglich zu machen und Fehlerquellen auszuschließen,
2. die vollständige Erfassung aller publizierten Artikel ohne das Risiko, Artikel zu verpassen,
3. eine rückwirkende Erfassung von Artikeln – oder zumindest die Möglichkeit, Artikel über lange Zeiträume automatisch zu erheben, ohne dass ein manuelles Eingreifen nötig ist,
4. ein Output in Form (a) eines standardisierten Datensatzes mit unter anderem Erscheinungszeitpunkt, URL, Titel und ggf. weiterer Metadaten sowie (b) die Speicherung der Artikel im Volltext (ohne irrelevante Elemente wie Navigationselemente oder Werbung) zur Archivierung und weiteren manuellen oder automatischen Analyse.

2. Ein Verfahren zur automatisierten Datenerhebung

Im Folgenden wird erläutert, wie die unter 1.5 vorgestellten Punkte umgesetzt werden können. Anstelle von in Großprojekten sinnvollen und nötigen High-End-Verfahren werden hier Bausteine für eine Methode vorgestellt, die auch ohne Programmierkenntnisse angewendet werden kann. Sie eignet sich somit nicht nur für größere Projekte, sondern kann selbst für studentische (Abschluss-) Arbeiten verwendet werden.

2.1 RSS-Feeds als Lösungsansatz

Ein Nutzer, der eine Nachrichtenseite lesen will, surft für gewöhnlich zu dieser Seite und fragt dort die gewünschte Information ab. Offenkundig ist es für unsere Ziele von Vorteil, wenn neue Beiträge automatisch zusammengestellt werden und die Seite nicht auf gut Glück auf neue Beiträge überprüft werden muss. Ein solches Vorgehen ermöglichen sogenannte RSS-Feeds. Dieses Verfahren wird unter Bezeichnungen wie „Dynamisches Lesezeichen“ auch genutzt, um in Browsern die neuesten Nachrichten von beispielsweise CNN, SpiegelOnline oder der Tagesschau anzuzeigen (Abb. 1).

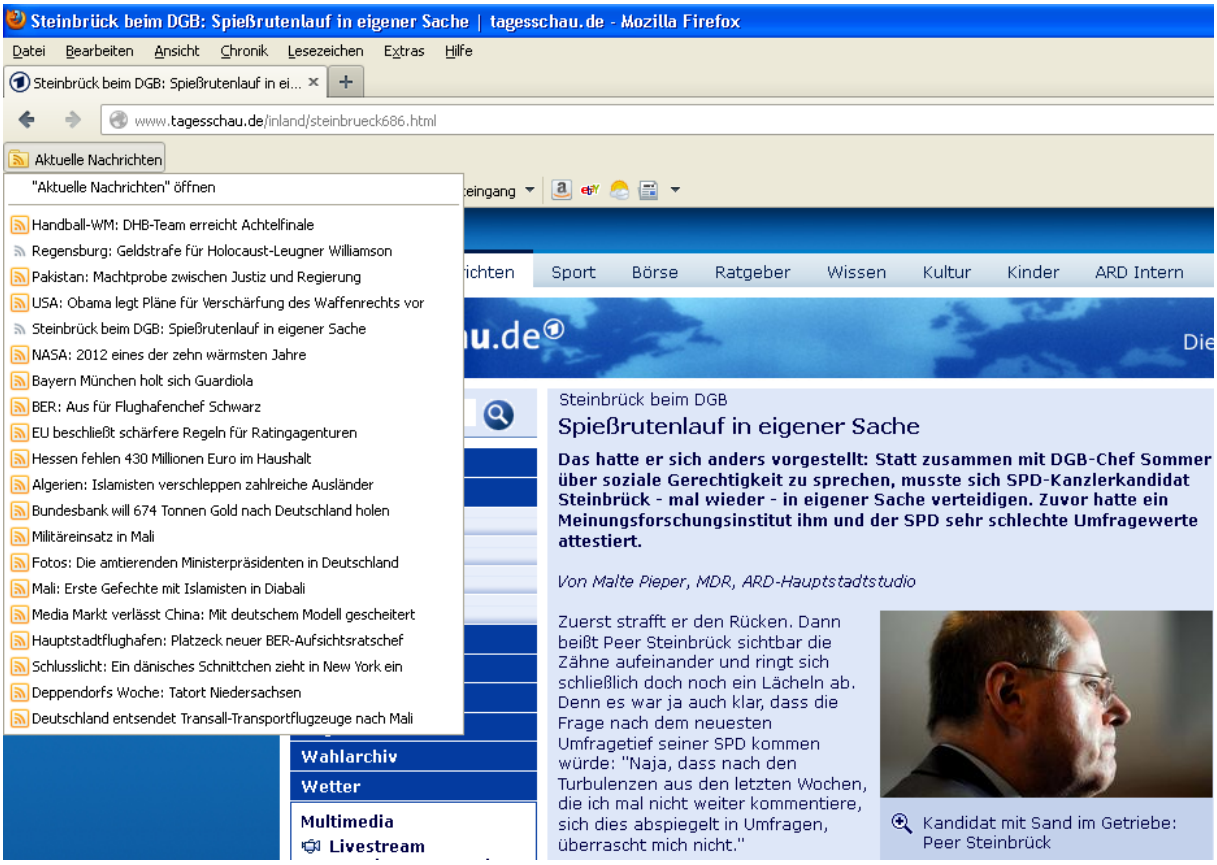

Abbildung 1. Eine populäre RSS-Anwendung: Die „Aktuellen Nachrichten“ im Browser Firefox.

Für den technisch Interessierten existieren zahlreiche Handbücher, die die Möglichkeiten des Formats vorstellen und praktische Anweisungen zur Programmierung bieten (z.B. Johnson, 2006). Auch ist es möglich, RSS-Feeds zu personalisieren oder Suchanfragen an Feeds umzuleiten (Bakker & Bakker, 2011). Solche weiterführenden Möglichkeiten werden im Rahmen dieses Beitrags jedoch nicht diskutiert. Für unsere Zwecke reicht ein basales Verständnis der groben Struktur eines RSS-Feeds vollkommen aus.

```
<item>
<title>Umbau beim VfL Wolfsburg: Magaths Spätlese</title>
<link>http://www.spiegel.de/sport/fussball/0,1518,824556,00.html#ref=rss</link>
<description>Plötzlich träumen sie in Wolfsburg wieder von der Europa League. Nach drei Siegen in Folge scheint die Mannschaft von Felix Magath endlich so etwas wie eine Struktur zu haben. Hat sich der Dauerumbau der vergangenen Monate also doch gelohnt?</description>
<pubDate>Sat, 31 Mar 2012 14:12:18 +0200</pubDate>
<guid>http://www.spiegel.de/sport/fussball/0,1518,824556,00.html</guid>
<content:encoded><![CDATA[Plötzlich träumen sie in Wolfsburg wieder von der Europa League. Nach drei Siegen in Folge scheint die Mannschaft von Felix Magath endlich so etwas wie eine Struktur zu haben. Hat sich der Dauerumbau der vergangenen Monate also doch gelohnt?]]></content:encoded>
</item>
```

Abbildung 2. Ausschnitt aus einem RSS-Feed.

Bei RSS-Feeds handelt es sich um ein standardisiertes Format, das seitenübergreifend (annähernd) gleich ist und aus den gleichen Elementen besteht: Ein RSS-Feed von SpiegelOnline unterscheidet sich also nicht grundlegend von dem eines beliebigen Blogs. Ein solches Format lässt sich relativ problemlos automatisiert auswerten. Entsprechend hat beispielsweise Erlhofer (2010) auf die Vorteile der Nutzung von RSS-Feeds bei der Erfassung von Blogbeiträgen hingewiesen. Auch die meisten Nachrichtenseiten bieten ihre Nachrichten in diesem Format an (siehe z.B. Greer & Yan, 2011). Um die Abonnenten dennoch auf die eigene Website zu locken, publizieren kommerzielle Nachrichtensites im Gegensatz zu vielen Blogs an Stelle des Volltextes meist lediglich die Schlagzeile und einen Teaser – in der Regel den ersten Absatz des Artikels – sowie einen Link zum Volltext auf der Website. Dieses Problem lässt sich jedoch, wie im Abschnitt 2.2.3 gezeigt wird, teilweise auffangen, indem man mit einem geeigneten Programm diese Links automatisiert herunterladen lässt.

2.2 Praktische Umsetzung

Es bestehen diverse Möglichkeiten, solche Feeds zu erfassen. Die hier vorgestellten Programme und Routinen sind daher als Beispiele zu verstehen.
Konkret werden drei Schritte beschrieben, die für eine systematische, transparente und reproduzierbare Inhaltsanalyse nötig sind:

1. Abonnieren der RSS-Feeds,
2. Erstellen eines Datensatzes,
3. Archivieren der Artikel.

2.2.1 Abonnieren der RSS-Feeds

Um die RSS-Feeds abonnieren zu können, sind zwei Dinge nötig: Erstens ein Link zu dem Feed, der häufig im ‚Kleingedruckten' der Website verborgen ist oder sich hinter dem Symbol  verbirgt, und zweitens einen sogenannten Feedreader, ein Programm zum Lesen der Feeds. Theoretisch könnte man die Feeds auch von Hand abspeichern, verlöre dann aber den Vorteil der automatischen Updates – man müsste selbst regelmäßig die Feeds überprüfen und die Datensätze zusammenführen. Eine Vielzahl von Feedreadern ist auf dem Markt. Die Entscheidung für die Verwendung eines bestimmten Readers richtet sich im Wesentlichen nach persönlichen Präferenzen.

Für unser Anliegen ist es jedoch wichtig, dass der Feedreader die Feeds nicht nur lesen, sondern auch wieder exportieren kann. Das ist jedoch häufig nicht der Fall.

Die beste Lösung hierzu bot lange Zeit der GoogleReader (http://reader.google.com), bei dem dieses Problem nicht bestand und der zudem einen weiteren Vorteil hatte: Das Programm musste nicht auf einem Rechner installiert werden, sondern lief online. Es war also nicht nötig, das Programm selbstständig in gewissen Abständen zu starten und zum Downloaden der Feeds zu veranlassen. Ab dem Moment, in dem ein Feed abonniert wurde, wurden sämtliche Beiträge des Feeds durch den Server von Google regelmäßig abgerufen und im GoogleReader-Account des Nutzers hinterlegt. Durch die Eingabe einer bestimmten URL (siehe hierzu "GoogleReaderAPI," n.d., und "Using the Google Reader API," n.d.) war es möglich, diese Beiträge allesamt in einem für die weitere Analyse geeigneten Format herunterzuladen. Die Nutzbarkeit dieser Schnittstelle, einer sogenannten API, für die kommunikationswissenschaftliche Forschung wurde im Rahmen einer Inhaltsanalyse österreichischer Online-Nachrichtenmedien erprobt, die eine automatische mit einer manuellen Codierung verband (Trilling & Schoenbach, 2012) und wurde außerdem in mehreren studentischen Projekten angewandt.

Am 14. März 2013 gab Google jedoch völlig überraschend bekannt, zum 1. Juli 2013 den Dienst einzustellen. Zum jetzigen Zeitpunkt ist kein anderer webbasierter Feedreader auf dem Markt, der eine solche offene und dokumentierte API bereitstellt. Diese einfache Möglichkeit, Daten für kommunikationswisschenschaftliche Inhaltsanalysen zu erheben, geht also verloren. Der in der Netzgemeinde als inoffizieller GoogleReader-Nachfolger gehandelten Dienst feedly.com hat angekündigt, die GoogleReader-API nachzubauen (Bager & Baum, 2013). Es ist jedoch noch nicht möglich, die API direkt aufzurufen und Items der abonnierten Feeds in eine Datei zu exportieren. Allerdings existiert ein Workaround: Der Dienst If This Then That (www.ifttt.com) ermöglicht es, feedly mit einem Cloud-Speicher wie Dropbox oder einem virtuellen Notizbuch wie Evernote zu verbinden. Auf diese Weise kann das Erscheinen eines neuen Eintrags aufgezeichnet werden. Eine Tabelle mit allen Artikeln inklusive ihrer Metadaten lässt sich so jedoch nicht erstellen. Für viele Anwendungszwecke entfällt daher bis auf Weiteres die Möglichkeit, einen webbasierten RSS-Reader zu nutzen. Zum jetzigen Zeitpunkt lassen sich drei Alternativen unterscheiden:

(1) *Ein Feedreader auf dem eigenen Rechner.* Eine naheliegende Lösung ist, einen der zahlreichen Reader auf dem eigenen Rechner zu installieren. Hierbei muss darauf geachtet werden, das Programm regelmäßig zu starten oder es besser noch automatisch täglich starten zu lassen, sodass die Feeds auch tatsächlich abgerufen werden. Größtes Problem ist, ein Programm zu finden, dass es ermöglicht, die gespeicherten Feeds in einem Format zu exportieren, das zur Weiterverarbeitung geeignet ist. Zwar wäre es auch denkbar, einen Codierer die Feeds direkt im Feedreader lesen zu lassen und auf eine Exportmöglichkeit zu

verzichten, jedoch wäre dieses Vorgehen im Hinblick auf die Archivierung des Untersuchungsmaterials problematisch. Das Programm RSSOwl ermöglicht den Export der Feeditems als HTML-Datei – und eigentlich auch als XML-Datei, wobei letzteres bei Tests des Autors auf verschiedenen Plattformen nicht zuverlässig funktionierte. Das XML-Format ist jedoch das für eine vielseitige Weiterverarbeitung geeignetste Format. Sehr zu empfehlen ist daher der Feedreader rawdog, der mit dem archive-Plugin den Export ins XML-Format ermöglicht. Für den nicht-technikaffinen Nutzer ist allerdings die Steuerung des Programms über die Kommandozeile gewöhnungsbedürftig. Andererseits bietet die Kommandozeilensteuerung große Vorteile, wenn das Programm automatisch gesteuert werden soll. Auch ist es mit einigem technischen Verständnis möglich, rawdog so anzupassen, dass die Artikel in einem selbst definierten Format ausgegeben werden.

(2) *Der selbstgeschriebe*ne *RSS-Client.* Eine weitere Möglichkeit,besteht darin, selbst ein Programm zu schreiben, das die Feeds abruft und gleich in der für das konkrete Forschungsprojekt benötigten Form aufbereitet. Für die Programmiersprache Python existiert die fertige Bibliothek feedparser[2], die einen Großteil der nötigen Aufgaben übernimmt. Denkbar wäre ein System, das die Feeds herunterlädt, archiviert, und direkt an ein System zur automatischen Inhaltsanalyse weiterreicht. Auch wenn es sich bei Python um eine verhältnismäßig leicht zu erlernende Sprache handelt, würde eine detaillierte Beschreibung im Rahmen dieses Beitrags zu weit führen.

(3) *Der eigene Server*. Man entscheidet sich als Forschungsgruppe, selbst die Rolle von Google zu übernehmen und einen Server zu betreiben, auf dem die RSS-Feeds gesammelt werden. Mitarbeiter können dann über das Internet auf diesen Server zugreifen. Für langfristige Projekte ist dies eine attraktive Möglichkeit, erfordert aber einen hohen organisatorischen und technischen Aufwand.

Für welche Methode man sich auch entscheidet, es empfiehlt sich, frühzeitig alle möglicherweise einmal relevant werdenden Feeds zu abonnieren und die häufig in Feedreadern aktivierte Option, alte Daten zu entfernen, zu deaktivieren. So kann auch später immer auf eine Vollerhebung aller Items eines Mediums seit Beginn des Abonnements zurückgegriffen werden. Damit lässt sich das Problem der nachträglichen Erfassung zwar nicht gänzlich lösen, aber doch vermindern, wenn die Feeds über einen langen Zeitraum sozusagen vorsorglich erfasst werden: Es ist in jedem Fall nicht mehr nötig, den Artikel persönlich abzurufen, während er online steht. Da es sich um Textdateien handelt, ist der Speicherbedarf zu vernachlässigen.

[2] Die Dokumentation des Pakets findet sich unter http://pythonhosted.org/feedparser/ , ein Anwendungsbeispiel unter http://www.pythonforbeginners.com/python-on-the-web/using-feedparser-in-python/

2.2.2 Erstellen eines Datensatzes

Für die nun folgende Erläuterung der Erstellung eines Datensatzes gehen wir davon aus, dass im vorigen Schritt eine XML-Datei mit allen zu analysierenden Items erstellt wurde. Während wiederum unzählige Möglichkeiten existieren, diese XML-Dateien in andere Formate zu konvertieren, dürfte die für viele Nutzer weitaus unproblematischste Lösung sein, das vermutlich ohnehin vorhandene Microsoft Excel zu nutzen: Neuere Versionen von Excel für Windows (nicht jedoch für Mac!) können XML-Dateien relativ problemlos lesen. Neben einer Anzahl irrelevanter Spalten erhält die Tabelle Spalten mit Veröffentlichungsdatum, URL, Autor, Titel, Text und weiteren Informationen. Im Textfeld ist bei großen Nachrichtensites in der Regel der Teaser des Artikels abgelegt, bei Blogs häufig sogar der gesamte Blogbeitrag.

Es empfiehlt sich, den so gewonnen Datensatz zu bereinigen, bevor er einem Format zur Weiterverarbeitung gespeichert wird: Redundante Spalten sollten entfernt werden und die Spalten sollten mit aussagekräftigen (Variablen-)Namen versehen werden. Außerdem kann es je nach Erkenntnisinteresse sinnvoll sein, innerhalb der Text-Spalte die HTML-Tags zu entfernen, indem in dieser Spalte über die Suchfunktion der String <*> durch ein leeres Feld ersetzt wird. Dies erhöht zum einen die Lesbarkeit des Textes, begünstigt zum anderen aber auch eine spätere automatische Analyse. Auch könnte man lediglich Formatierungstags (wie <b> für Fettdruck) entfernen, und eingebundene Links oder Abbildungen erhalten. Da die ursprüngliche XML-Datei zusätzlich zur in diesem Schritt erstellten Arbeitsdatei erhalten bleibt, kann ein solcher Eingriff bei Bedarf rückgängig gemacht werden.

Falls angestrebt wird, die erstellte Excel-Datei in einem anderen Programm – beispielsweise einem Statistikprogramm – weiterzuverarbeiten, so sollte man darauf achten, den Zelltyp insbesondere der Teaser-Spalte über den Befehl „Zellen formatieren“ auf „Text“ zu setzen, da ansonsten längere Texte beim Ex- oder Import möglicherweise nicht korrekt übertragen werden, sondern durch eine Folge von #-Zeichen ersetzt werden.

Im Falle einer manuellen Codierung bietet es sich an, den Codierern eine so bereinigte Version dieses Datensatzes, ergänzt um Felder für die zu codierenden Variablen, zur Verfügung zu stellen. Sie können dann über die enthaltenen Links direkt auf die zu codierenden Materialien zugreifen und die Codierung vornehmen – wobei natürlich auch bei einer solchen Vorgehensweise die Sicherung und Archivierung nicht vernachlässigt werden darf. Man könnte die Codierer nun instruieren, den Beitrag beim Codieren abzuspeichern – oder man verwendet, wie im nächsten Abschnitt vorgestellt, eine automatisierte Methode.

Alternativ könnte man erwägen, die XML-Datei über frei erhältliche Kommandozeilentools in ein CSV-Format zu überführen, um so erstens den Einsatz der kostenpflichtigen Software Excel zu umgehen und zweitens den gesamten Prozess der Datenerhebung noch weitgehender automatisieren zu können. In sehr großen Forschungsprojekten könnte auch ein selbst angepasster XML-Parser (ein Script, das eine XML-Datei auswertet) sinnvoll sein, um die nötigen Daten direkt aus der XML-Datei auszulesen.

2.2.3 Archivieren der Artikel

Je nach Anwendungsgebiet sind die im RSS-Feed enthaltenen Texte für die Codierung bereits ausreichend; denn während wie oben erläutert große Nachrichtensites meist nur den ersten Absatz im RSS-Feed veröffentlichen und einen Link zum Volltext auf ihrer Website bereitstellen, sind diese Teaser bei Blogs meist sogar völlig identisch mit dem Volltext des Beitrags. Eine separate Archivierung der Artikel, auf die sich der jeweilige RSS-Feed bezieht, kann dann häufig entfallen.
Für alle anderen Fälle können die URLs der Beiträge an ein Downloadprogramm übergeben werden: Sie liegen bislang in einer Spalte des im vorigen Schrittes erstellten (Excel-)Datensatzes vor. Im Falle des kostenlosen Kommandozeilenprogramms wget reicht es beispielsweise, diese Spalte in eine Textdatei zu kopieren und abzuspeichern (z.B. unter dem Namen url.txt) und dann wget aufzurufen (durch Eingabe des Befehls wget –i url.txt in der Kommandozeile des Betriebssystems). Über weitere Parameter kann eingestellt werden, ob von dem Beitrag ausgehend weiterführende Links ebenfalls heruntergeladen werden sollen. Auch stehen Optionen bereit, um Websites, die das automatisierte Downloaden zu unterbinden versuchen, zu überlisten, indem ein manuelles Downloaden suggeriert wird, beispielsweise durch das Einbauen von zufälligen Wartezeiten. Die gleiche Aufgabe kann auch von einem Python-Script erledigt werden, was den Vorteil hat, dass im gleichen Arbeitsschritt auch unerwünschten Bestandteile wie Werbung oder Navigationselemente entfernt werden können und zudem eine erste automatische Analyse erfolgen kann. Ein Beispiel für ein solches Script kann auf der Website des Autors heruntergeladen werden[3].

3. Fazit

Im Hinblick auf den aufgestellten Anforderungskatalog (wenig manuelle Schritte, vollständige Erfassung, standardisierter Output und Archivierung) lässt sich feststellen, dass die vorgeschlagene

[3] http://www.damiantrilling.net/tools/

Methode die Anforderungen erfüllt. Ein RSS-basiertes Vorgehen ermöglicht das Erfassen verschiedenster Onlinemedien, verringert Fehlerquellen und kann mit vergleichsweise geringem Aufwand durchgeführt werden. Wenn ein handelsüblicher Feedreader eingesetzt wird und die Items manuell codiert werden, sind keinerlei besondere technische Kenntnissen notwendig. Damit eignen sich RSS-basierte Erhebungsmethoden insbesondere auch für die zahlreichen kleineren Fallstudien und Abschlussarbeiten in unserem Fach. Mit entsprechenden Kenntnissen ist das Verfahren noch weiter ausbaufähig und kann weitgehend automatisiert und an eigene Analysewünsche angepasst werden.

In der vorgestellten Form ist das Verfahren vor allem auf Nachrichtenwebsites und Blogs zugeschnitten, zudem wird implizit davon ausgegangen, dass das Erkenntnisinteresse dem Text gilt. Nichts spricht jedoch dagegen, einzelne Teile des Verfahrens anzupassen und andere Teile zu übernehmen, um beispielsweise die Linkstruktur, die Verwendung von Multimediamaterial, das Vorhandensein von Kommentaren (die gelegentlich über eigene RSS-Feeds verfügen) oder andere Aspekte zu untersuchen.

Darüber hinaus soll der Beitrag auch als eine Anregung verstanden werden, die einzelnen Komponenten des Datenerhebungsprozesses von Online-Nachrichtenmedien weiterzuentwickeln und der Disziplin zur Verfügung zu stellen. So wären die nahtlose Integration der am Ende dieses Beitrags angerissenen Analysemöglichkeiten und ihre Anbindung an die automatisierte Erfassung wünschenswerte Schritte. Insbesondere die Entwicklung eines Systems, das die Erfassung, automatische und manuelle Codierung sowie die Analyse vereint, könnte nicht nur die Nutzbarkeit erhöhen, sondern auch zur Weiterentwicklung und Standardisierung der Online-Inhaltsanalyse beitragen.

Anhang

Übersicht der erwähnten Tools zur Nutzung von RSS-Feeds in der Inhaltsanalyse

Name	*Wo erhältlich?*	*Beschreibung*
RSSOwl	http://www.rssowl.org (Open Source)	RSS-Feedreader für Windows, MacOS und Linux. Leicht zu bedienen, keinerlei technische Kenntnisse nötig, viele Funktionen. Items können als HTML-Datei exportiert werden.
rawdog	http://offog.org/code/rawdog.html (Open Source)	Kommandozeilen-Feedreader. Benötigt Python (eine Programmiersprache, bei Linux und

		MacOS vorhanden, für Windows kostenlos erhältlich). Bietet Plugin zum XML-Export. Kann mit einigen technischen Kenntnissen weitreichend auf die konkreten Bedürfnisse zugeschnitten werden.
feedly	http://feedly.com	Online-Feedreader. Kann (in Grenzen) mit Tools wie If this then that (http://ifttt.com) ausgelesen werden.
wget	Vorinstalliert bei Linux. Für Windows: http://gnuwin32.sourceforge.net/packages/wget.htm Für MacOS: Eine Google-Suche nach „wget MacOS“ liefert zahlreiche Möglichkeiten (Open Source)	Kommandozeilentool zum automatischen Herunterladen von Webseiten. Beispielsweise können die im RSS-Feed eingebetteten URLs zum Volltext der Items an wget übergeben werden, um das Untersuchungsmaterial zu archivieren.

Literatur

Bager, J., & Braun, H. (2013). Nach dem Aus für Google Reader: Alternativen für RSS-Reader. *c't*. Retrieved from http://www.heise.de/ct/artikel/Nach-dem-Aus-fuer-Google-Reader-Alternativen-fuer-RSS-Reader-1823492.html

Bakker, T., & Bakker, P. (2011). *Handboek nieuwe media. Digitale technologieën begrijpen en gebruiken*. Alphen aan den Rijn, Netherlands: Kluwer.

Bröker, E. (1981). *Computerunterstützte Inhaltsanalyse der internationalen Berichterstattung: Zur Möglichkeit der systematischen Beschreibung internationaler Beziehungen*. Dissertation, Westfälische Wilhems-Universität Münster.

Erlhofer, S. (2010). Datenerhebung in der Blogosphäre: Herausforderungen und Lösungswege. In M. Welker & C. Wünsch (Eds.), *Die Online-Inhaltsanalyse. Forschungsobjekt Internet.* (pp. 144–166). Cologne: Herbert von Halem.

Früh, W. (2007). *Inhaltsanalyse: Theorie und Praxis*. Konstanz: UVK.

Greer, J. D., & Yan, Y. (2011). Newspapers connect with readers through multiple digital tools. *Newspaper Research Journal, 32*(4), 83–97.

GoogleReaderAPI. (n.d.). Retrieved January 13, 2013, from http://code.google.com/p/pyrfeed/wiki/GoogleReaderAPI

Günther, E. & Scharkow, M. (2012, September). Automatisierte Datenbereinigung bei Inhalts- und Linkanalysen von Online-Nachrichten. *Jahrestagung der Fachgruppe Methoden der Publizistik- und Kommunikationswissenschaft der DGPuK*. Zürich.

Johnson, D. (2006). *RSS and Atom in action*. Greenwich, CT: Manning.

Krippendorf, K. (2004). *Content analysis: An introduction to its methodology* (2nd ed.). Thousand Oaks, CA: Sage.

Lim, J. (2010). Convergence of attention and prominence dimensions of salience among major online newspapers. *Journal of Computer-Mediated Communication*, *15*(2), 293–313. doi:10.1111/j.1083-6101.2010.01521.x

Marcotte, E. (2010). Responsive Web Design. *A List Apart, 306.* Retrieved August 4, 2013 from http://alistapart.com/article/responsive-web-design

McMillan, S. J. (2000). The microscope and the moving target: The challenge of applying content analysis to the World Wide Web. *Journalism & Mass Communication Quarterly*, *77*(1), 80–98. doi:10.1177/107769900007700107

Meier, S., Wünsch, C., Pentzold, C., & Welker, M. (2010). Auswahlverfahren für Online-Inhalte. In M. Welker & C. Wünsch (Eds.), *Die Online-Inhaltsanalyse. Forschungsobjekt Internet.* (pp. 103–123). Cologne: Herbert von Halem.

Merten, K. (1995). *Inhaltsanalyse. Einführung in Theorie, Methode und Praxis* (2nd ed.). Opladen: Westdeutscher Verlag.

Quandt, T. (2008). Neues Medium, alter Journalismus? Eine vergleichende Inhaltsanalyse tagesaktueller Print- und Online-Nachrichtenangebote. In T. Quandt & W. Schweiger (Eds.), *Journalismus Online - Partizipation oder Profession?* (pp. 131–155). Wiesbaden: VS.

Rössler, P. (2005). *Inhaltsanalyse*. Konstanz: UVK.

Rössler, P., & Wirth, W. (2001). Inhaltsanalysen im World Wide Web. In W. Wirth & E. Lauf (Eds.), *Inhaltsanalyse: Perspektiven, Probleme, Potentiale* (pp. 280–302). Cologne: Herbert von Halem.

Rüdiger, K., & Welker, M. (2010). Redaktionsblogs deutscher Zeitungen. Über die Schwierigkeiten diese inhaltsanalytisch zu untersuchen – ein Werkstattbericht. In M. Welker & C. Wünsch (Eds.), *Die Online-Inhaltsanalyse. Forschungsobjekt Internet.* (pp. 448–468). Cologne: Herbert von Halem.

Schweitzer, E. J. (2010). Politische Websites als Gegenstand der Online-Inhaltsanalyse. In M. Welker & C. Wünsch (Eds.), *Die Online-Inhaltsanalyse. Forschungsobjekt Internet.* (pp. 44–102). Cologne: Herbert von Halem.

Sjøvaag, H., & Stavelin, E. (2012). Web media and the quantitative content analysis: Methodological challenges in measuring online news content. *Convergence: The International Journal of Research into New Media Technologies*, *18*(2), 215–229. doi:10.1177/1354856511429641

Trilling, D., & Schoenbach, K. (2012, June). How content fragmentation can increase audience fragmentation: Do people really expose themselves only to content they like? *World Association of Public Opinion Research Conference,* Hongkong.

Using the Google Reader API. (n.d.). Retrieved January 13, 2013, from http://blog.martindoms.com/2009/08/15/using-the-google-reader-api-part-1/

Zeller, F., & Wolling, J. (2010). Struktur- und Qualitätsanalyse publizistischer Onlineangebote. *Media Perspektiven*, (3/2010), 143–153.