Technical Report
KN–2014–DISY–02

Webharvesting von Publikationsdaten

Thomas Zink† **Oliver Haase†** **Marcel Waldvogel‡**

† Lehrstuhl Software Engineering und Verteilte System
Hochschule Konstanz Technik, Wirtschaft und Gestaltung – Germany

‡ Lehrstuhl Verteilte Systeme
Universität Konstanz – Germany

Diese Arbeit wurde unterstützt durch die IQF (Innovations- und Qualitätsfonds) Förderlinie des Landes Baden-Württemberg im Rahmen des Projektes ''Kooperative und nachhaltige Archivierung von Webinhalten an Hochschulen'' .

Zusammenfassung Forschungsarbeiten, -daten und -resultate an Universitäten und Hochschulen werden immer häufiger nicht mehr als Schriftstück, sondern exklusiv auf Webseiten im Internet und Intranet veröffentlicht und dokumentiert. Diese werden bisher nur ungenügend und unvollständig archiviert. Dadurch entstehen potentiell große Lücken in der Archivierung und künftigen Dokumentation. Zudem beeinträchtigt dies die Nachvollziehbarkeit und Reproduzierbarkeit, beides Eigenschaften, die besonders im wissenschaftlichen Kontext einen hohen Stellwert haben.

Keywords. *Web-Archivierung, harvesting, crawling, Publikationen*

Inhaltsverzeichnis

1 Einleitung

Beim Harvesting von Publikationsdaten handelt es sich um sogenanntes *fokussiertes crawling* (engl. *focused crawling*), d.h. die gezielte Suche nach Inhalten, die bestimmte Eigenschaften erfüllen. Hierbei werden Webseiten gecrawlt und auf diese Eigenschaften untersucht. Solche Ressourcen, die die gewünschten Eigenschaften besitzen, werden geharvested. Das fokussierte Crawling beinhaltet selbst noch nicht das Extrahieren von Metadaten.

Die Extraktion von Metadaten ist wiederum ein eigenes Problem und gliedert sich in mehrere Teilprobleme, denn die Metadaten können an unterschiedlichen Stellen und in unterschiedlichen Formaten auftauchen. Zum einen sind Publikations-Metadaten in Publikationen selbst enthalten, zum Anderen sind sie in Referenzen auf Publikationen enthalten. Metadaten können also in unterschiedlichen Dokumenttypen und in unterschiedlichen Formaten vorkommen.

Abbildungen 1 bis 4 zeigen die Metadaten einer Publikation in Unterschiedlichen Formaten und Dokumenten.

Rentrop, Christopher; Zimmermann, Stephan: Shadow IT Evaluation Model; In: Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS), Wrocław, Poland, 9 - 12 September, 2012, pp. 1023–1027.

Abbildung 1. Referenz auf eine Publikation in der Referenzliste der Homepage eines Autors.

Proceedings of the Federated Conference on Computer Science and Information Systems pp. 1023–1027

ISBN 978-83-60810-48-4

Shadow IT Evaluation Model

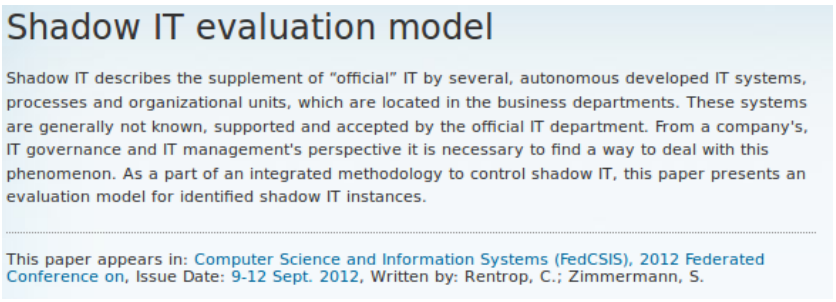
Christopher Rentrop
HTWG Konstanz – University of Applied Sciences
Brauneggerstr. 55, 78462 Konstanz, Germany
Email: rentrop@htwg-konstanz.de

Stephan Zimmermann
HTWG Konstanz – University of Applied Sciences
Brauneggerstr. 55, 78462 Konstanz, Germany
Email: stephan.zimmermann@htwg-konstanz.de

Abbildung 2. Metadaten im PDF Dokument der Publikation.

Die Metadaten einer Veröffentlichung sind in dem veröffentlichten Dokument selbst in Form von Titel, Autor, etc vorhanden. Liegen die Publikationen selbst in Form von Dokumenten vor, ist es daher möglich, die Metadaten direkt aus dem Dokument zu extrahieren. Hierzu muss zuerst der Text aus der Publikation extrahiert werden. Der extrahierte Text kann dann auf Metadaten untersucht werden.

Oftmals sind wissenschaftliche Publikationen jedoch an unterschiedlichen Stellen - wie Verlagen, Portale der Fachzeitschriften oder Konferenzen - veröffentlicht und auf den Publikationsseiten einer Organisation (Hochschule) lediglich referenziert und nicht direkt zugänglich. In solchen Fällen hat man keinen

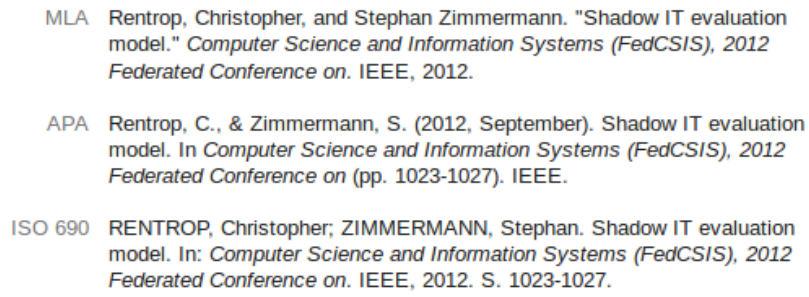


Shadow IT evaluation model

Shadow IT describes the supplement of "official" IT by several, autonomous developed IT systems, processes and organizational units, which are located in the business departments. These systems are generally not known, supported and accepted by the official IT department. From a company's, IT governance and IT management's perspective it is necessary to find a way to deal with this phenomenon. As a part of an integrated methodology to control shadow IT, this paper presents an evaluation model for identified shadow IT instances.

This paper appears in: Computer Science and Information Systems (FedCSIS), 2012 Federated Conference on, Issue Date: 9-12 Sept. 2012, Written by: Rentrop, C.; Zimmermann, S.

Abbildung 3. Metadaten im HTML-Dokument der Publikation.



MLA Rentrop, Christopher, and Stephan Zimmermann. "Shadow IT evaluation model." *Computer Science and Information Systems (FedCSIS), 2012 Federated Conference on*. IEEE, 2012.

APA Rentrop, C., & Zimmermann, S. (2012, September). Shadow IT evaluation model. In *Computer Science and Information Systems (FedCSIS), 2012 Federated Conference on* (pp. 1023-1027). IEEE.

ISO 690 RENTROP, Christopher; ZIMMERMANN, Stephan. Shadow IT evaluation model. In: *Computer Science and Information Systems (FedCSIS), 2012 Federated Conference on*. IEEE, 2012. S. 1023-1027.

Abbildung 4. Verschiedene Zitationsstile bei Google Scholar.

Zugriff auf die Publikation selbst, sondern nur auf eine Referenz, die Metadaten wie Autor, Titel, Jahr, etc enthält, und wiederum in unterschiedlichen Formaten, auch genannt Stilen, vorkommen können. Um Metadaten aus den Referenzen zu gewinnen, müssen diese geparst werden. Dieses Problem bezeichnet man als *citation parsing*. Hierbei muss der Stil der Referenz erkannt werden, um die einzelnen Felder zu extrahieren. Es ist also beschränkt einsetzbar auf bekannte Formate. Da Referenzen auf Webseiten in einer hohen Anzahl an unterschiedlichen Formaten auftauchen können, muss man sich auf gängige Formate beschränken. Referenzen, die diesen Formaten nicht folgen, werden somit nicht korrekt erkannt und geparst. Es ist aber natürlich möglich, Organisations-spezifische Formate zusätzlich zu integrieren, so fern diese bekannt sind.

Die extrahierten Metadaten können selbst wiederum in verschiedenen Formaten gespeichert werden. Gängige Bibliographie-Formate sind bspw Bibtex, RefMan, EndNote oder XML. Manchmal sind diese sogar in Literaturlisten bereits vorhanden oder verlinkt, allerdings kann man sich darauf nicht verlassen. Listings 1 bis 4 zeigen Beispiele verschiedener Bibliographie-Formate zur Beispielpublikation aus Abbildung 1.

```
@inproceedings{rentrop2012shadow,
        title={Shadow IT evaluation model},
        author={Rentrop, Christopher and Zimmermann,
            Stephan},
```

```
        booktitle={Computer Science and Information Systems
            (FedCSIS), 2012 Federated Conference on},
        pages={1023--1027},
        year={2012},
        organization={IEEE}
}
```

Listing 1. Bibliographieeintrag in Bibtex.

```
TY  - CONF
T1  - Shadow IT evaluation model
A1  - Rentrop, Christopher
A1  - Zimmermann, Stephan
JO  - Computer Science and Information Systems (FedCSIS),
    2012 Federated Conference on
SP  - 1023
EP  - 1027
SN  - 1467307084
Y1  - 2012
PB  - IEEE
ER  -
```

Listing 2. Bibliographieeintrag in RefMan.

```
%0 Conference Proceedings
%T Shadow IT evaluation model
%A Rentrop, Christopher
%A Zimmermann, Stephan
%B Computer Science and Information Systems (FedCSIS), 2012
    Federated Conference on
%P 1023-1027
%@ 1467307084
%D 2012
%I IEEE
```

Listing 3. Bibliographieeintrag in EndNote.

```
<book ...>
        <rft:atitle>IT Evaluation Model</rft:atitle>
        <rft:spage>12</rft:spage>
        <rft:date>2012</rft:date>
        <rft:btitle>In: Proceedings of the Federated
            Conference on Computer Science and Information
            Systems (FedCSIS</rft:btitle>
        <rft:genre>proceeding</rft:genre>
        <rft:epage>1023</rft:epage>
        <rft:au>Rentrop Christopher</rft:au>
        <rft:au>Zimmermann Stephan Shadow</rft:au>
</book>
```

Listing 4. Bibliographieeintrag in FreeCite XML.

Das Extrahieren von Publikations-Metadaten in ein komplexes Problem, das mehrere Lösungsmöglichkeiten aus unterschiedlichen Bereichen der Informationsverarbeitung bietet und fordert. Im Folgen wird auf die einzelnen Lösungswege detailiert eingegangen.

2 Übersicht

Abstrahiert folgt die Extraktion von Metadaten immer folgendem Schema.

0. Beschaffung relevanter Dokumente, bspw. HTML, PDF, PostScript.
1. Extrahieren von Text aus einem Dokument (*text extraction*)
2. Parsen des Textes und identifizieren von Kandidaten (*document parsing*)
3. Extraktion der Metadaten aus den Kandidaten (*text/citation parsing*)

Die Beschaffung von Dokumenten zur Extraktion von semi-strukturierten Publikations-Metadaten aus unstrukturierten Web-Daten kann an unterschiedlichen Stellen erfolgen.

1. Live beim Harvesten der Webseiten. Hierbei wird jeder gefundene Link bzw jede geharvestete Seite direkt auf Publikationsdaten untersucht. Dies bezeichnet man als ''fokussiertes Crawling''.
2. Forensisch im Archiv. Dokumente werden aus den vom Harvester erstellten Archiven (warcs) extrahiert. Hierbei muüssen aus den Archiven die Webseiten und weitere relevante Dokumente wieder rekonstruiert werden.
3. Forensisch mit Hilfe des Logs. Hierbei werden die Log-Daten des Harvesters analysiert. URLs auf relevante Dokumente werden extrahiert. Diese werden anschliessend erneut abgerufen und auf Publikationsdaten hin untersucht.

Liegen die Dokumente vor, muss zunächst festgestellt werden, ob sie Publikationsdaten enthalten, d.h. es müssen Publikationsdaten bzw Referenzen identifiziert werden. Hierzu muss Text aus den Dokumenten extrahiert werden und dieser auf vorher definierte Eigenschaften untersucht werden. So gefundene Kandidaten können dann weiter analysiert und ggf. strukturiert werden. Dafür müssen die Kandidaten geparst werden. Das Parsing wird nicht für alle Kandidaten erfolgreich sein, daher erhält man als Ergebnis Listen von strukturierten und nicht strukturierten Publikationsdaten, die auf Webseiten gefunden wurden.

Die Suche in textbasierten Dokumenten wie HTML unterscheidet sich leicht zu der in binären Dokumenten wie PDF und PostScript.

Unter der Annahme es liegen HTML-Seiten vor, findet die Extraktion der Metadaten in folgenden Prozessen statt.

1. Identifizieren von Seiten / Dokumenten, die Publikationsdaten enthalten.
2. Identifizieren von Referenzen auf den gefundenen Publikationsseiten.
3. Extraktion der Metadaten durch *citation parsing*.

2.1 Extrahieren von Dokumenten

Grundlage zum Extrahieren von Publikationsdaten ist das Vorhandensein der Daten in Dokumenten mit definierten und spezifischen Formaten. Publikationen sowie Referenzen/Zitationen kommen vor allem in PDS, Postscript und HTML-Dokumenten vor, wobei binäre Formate hauptsächlich für Publikationen und HTML für Referenzlisten verwendet werden. Auch muss man unterscheiden zwischen den Referenzelisten eines Autors auf seiner Homepage, die eigene publizierte Werke referenziert, und der Referenzeliste in Publikationen, die zum großteil fremde Werke referenziert.

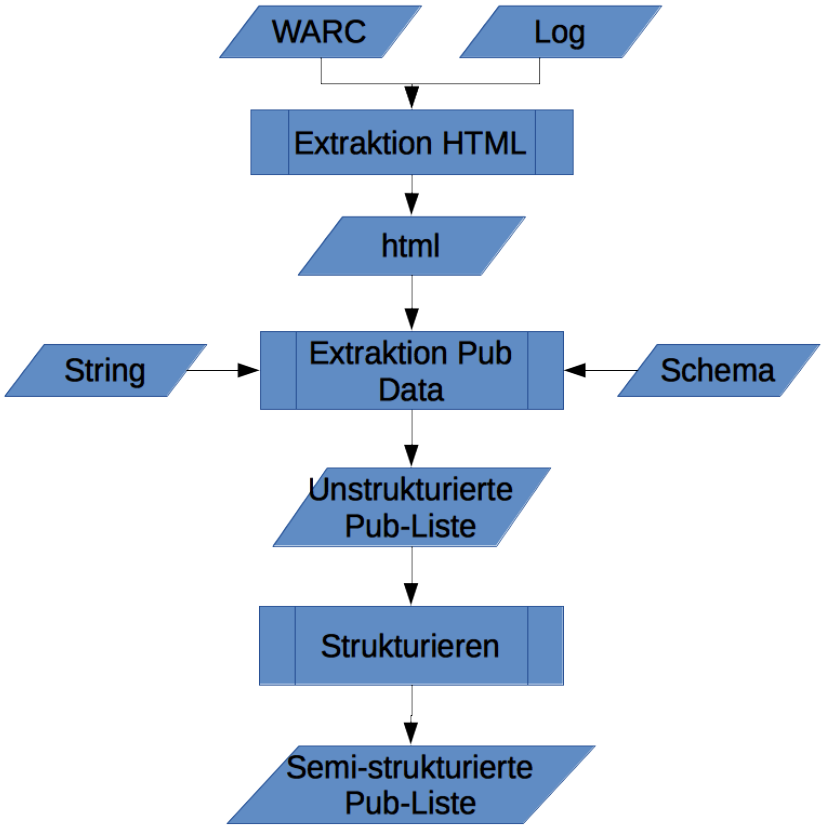


Abbildung 5. Extraktion von Metadaten.

In diesem Arbeitspaket liegt der Schwerpunkt auf der Extraktion strukturierter Publikationsdaten aus unstrukturierten Webdaten. Im Fokus steht daher das Erkennen und Strukturieren von Zitationen / Referenzen in Publikationslisten auf Webseiten von Autoren. Lösungen zum extrahieren von Metadaten aus binären Publikationen werden später ausführlich diskutiert.

Grundsätzlich bieten sich drei Alternativen an, um an die HTML-Seiten kommen. Diese sind im Abbildung 6 beschrieben.

Zum einen können die Seiten direkt beim Harvesten einer Domain an den Publikations-Extrakter weitergegeben werden. Der Vorteil ist, dass dadurch die Seiten direkt beim Harvesten auf Publikationsdaten hin untersucht werden können und keine weiteren Untersuchungen und keine weiteren Crawling-Läufe anfallen. Allerdings erfordert solch eine Lösung individuelle Anpassungen an eine Web-Harvesting-Software, was auch auf Grund der Notwendigkeit fortdauernder Anpassungen an neue Versionen unpraktikabel und teuer ist.

Eine weitere Möglichkeit ist daher das Extrahieren der HTML-Seiten aus den Archiven des Harvesters. Hierbei werden forensisch die Archive nach HTML-Seiten durchsucht und diese dann an den Publikations-Extraktor weitergegeben. Auch hierbei muss die gecrawlte Domain nicht erneut angesehen werden. Allerdings ist das Auspacken von und Suchen in Archiven ein nicht zu ver-

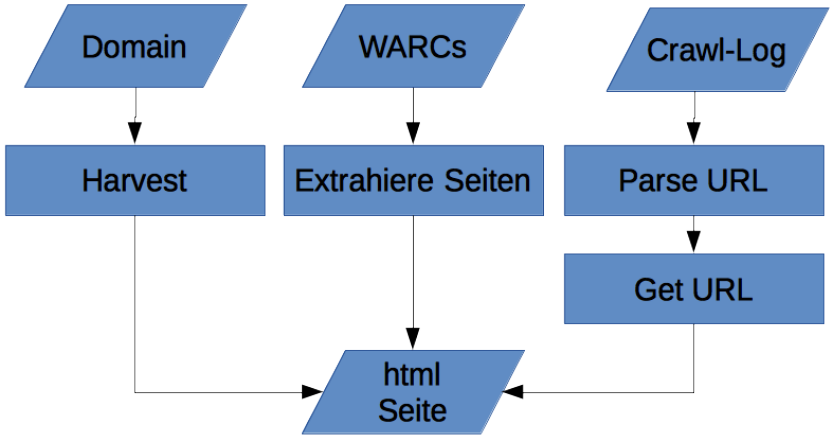


Abbildung 6. Extraktion von HTML-Seiten.

nachlässigender Aufwand, der durchaus einiges an Rechenaufwand und Speicher benötigt, was in Komplexität abhängig von der Größe der Archive ist. Bei Archivierungssystemen, die viele Domains crawlen und archivieren und daher u.U. ohnehin unter Last und Platzmangel leiden, ist diese Lösung daher nicht wünschenswert.

Eine nächste Möglichkeit ergibt sich durch eine Analyse der Crawl-Log-Dateien des Harvesters. Hier befinden sich Informationen über gecrawlte URLs sowie deren Datentypen. Somit lässt sich recht einfach eine Liste von URLs erstellen, die auf HTML-Seiten verweisen. Über diese können die Seiten dann geziehlt abgerufen und analysiert werden. Diese Lösung ist am wenigsten komplex, erfordert aber ein zusätzliches abrufen von HTML-Seiten, was zu erhöhter Last im Netz und auf den Ziel-Servern führt. Da allerdings nur gezielt HTML-Seiten aufgerufen werden, ist der Aufwand um einiges geringer als ein vollständiger Crawl. Auch sind die Vorteile einer solchen Lösung bestechend. Die Implementierung ist denkbar einfach und fast vollkommen unabhängig von Software von Drittherstellern. Ausnahme ist hier das Format der Crawl Log-Dateien, das sich aber vorrausslichtlich nicht häufig ändert. Dadurch ist die Lösung flexibler und sowohl der Entwicklungs- als auch der Pflegeaufwand extrem gering im Vergleich zu den anderen beiden Lösungen.

2.2 Extraktion von Publikationsdaten

Aus den HTML-Seiten müssen Publikationsdaten extrahiert werden. Dieser Prozess gliedert sich in mehrere Schritte (Abbildung 7).

Identifizieren von Publikationsseiten. Seiten mit Publikationen enthalten i.d.R. deutliche Anzeichen, die auf das vorhandensein einer Publikationsliste deuten. Da Publikationen öffentlich - und damit auch mittels Suchmaschinen - gefunden werden sollen, reicht oft schon die Suche nach Schlüsselwörtern wie “Publikation“ oder “Veröffentlichung“. Da eine Suche im Quelltext einer Seite jedoch auch uninteressante Elemente wie Menu-Einträge, Links, oder auch

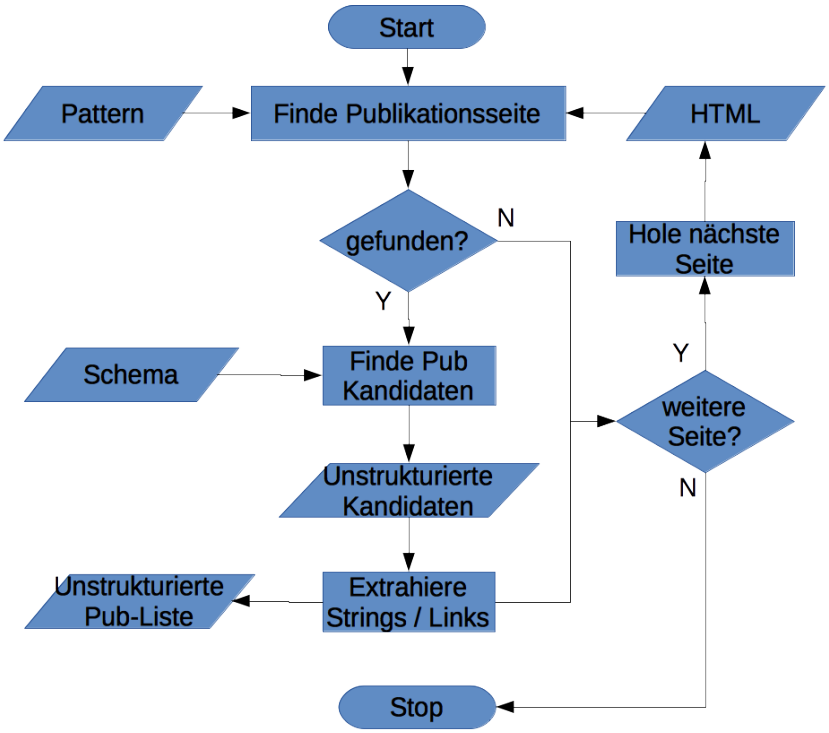


Abbildung 7. Extraktion von Publikationsdaten.

Script-Snippets liefert, muss etwas genauer differenziert werden. Die Suche nach den Schlüsselwörtern sollte daher auf aussagekräftige Webseiten-Elemente wie Titel und Überschriften beschränkt werden.

Listings 5 und 6 zeigen mögliche Implementierungen zur Identifikation von Publikationsseiten. Gesucht wird nach Titeln und Überschriften, die einen der Strings "Veröffentlichung", "Publikation" oder englisch "Publication" enthalten. Wird mindestens ein solches Element auf einer Seite gefunden, so wird diese als Publikationsseite betrachtet.

```
def has_publications(html):
        pattern = compile("<(title|h).*>.*(Ver(ö|&ouml)
            ffentlichung|Publikation|Publication).*</(title|
            h).*>")
        return (findall(pattern,html)) > 0
```

Listing 5. Identifikation von Publikationen in Python.

```
egrep "<((t|T)itle|H|h)[[:digit:]]*>(Ver(ö|&ouml)
    ffentlichung|Publikation|Publication)" [html page]
```

Listing 6. Identifikation von Publikationen in Bash.

So identifizerte Publikationsseiten werden in einem nächsten Schritt auf Referenz/Zitations-Listen hin untersucht.

Identifizieren von Publikationskandidaten Auf gefundenen Publikations-Seiten müssen einzelne Referenzen identifiziert und strukturiert werden. Da die Publikationsliste praktisch beliebig auf der Seite gestaltet werden kann und somit in einer großen Anzahl unterschiedlicher Formate vorliegen kann, ist ein scharfes Abgrenzen von Publikationsdaten nahezu unmöglich. Diese enorme Vielfalt an verschiedenen Formaten bedingt daher eine Einschränkung auf gängige und leicht erkennbare Stile. Hierdurch ist ein Verlust der Genauigkeit möglich.

Diesem Problem kann man unterschiedlich begegnen. Man kann Anwender dazu auffordern, Referenzen in gängigen Formaten zu veröffentlichen. Allerdings wird dies nur in wenigen Fällen erfolgreich sein. Alternativ können Kundenspezifische Regeln implementiert werden, die dort gängige Zitationsstile erfassen. Dies ist ein iterativer Prozess, da die Stile zu Beginn nicht unbedingt bekannt sind und diese erst durch Untersuchung der identifizierten Publikationsseiten auffallen.

Bibliographische Einträge sind oft eingebettet in eines der folgenden Elemente

- Paragraphen: $< p > Zitation < /p >$
- Listen: $< li > Zitation < /li >$
- Tabellen: $< tr > Zitation < /tr >$

Leider hat jedes dieser Elemente auch weitere, unterschiedliche Verwendungen. Paragraphen beinhalten auch anderen Text. Listen werden auch für Menus und Navigation verwedet. Publikationsdaten in Tabellen können beliebig komplex aufgebaut werden, z.b. gegliedert nach Autoren oder Jahren mit einzelnen Zeilen für unterschiedliche Angaben. Hier ist ein genaues Identifizieren ohne optische Analyse sehr schwierig.

Zusätzlich zu den Zitationen könnte man auch gezielt nach Links auf Dateien in gängigen Publikationsformaten (pdf, doc, bib) suchen. Der Link kann die Referenz selbst sein, oder ihr vor-/nachgestellt sein. Allerdings sind nicht alle Referenzen zwingend mit links ausgestattet und es ist nicht klar, ob, und in welchem Format bibliographische links vorhanden sind. Daher ist der Mehrwert eines solchen Vorgehens zweifelhaft, zumal Publikationen selbst gesondert eingesammelt werden können.

Um das Rauschen in den Publikationsdaten zu verringern ist es daher notwendig die Suche auf gängige Formate einzuschränken. Ein typischer bibliographischer Eintrag auf einer Webseite sieht bspw. wie folgt aus.

"Zimmermann, Stephan; Rentrop, Christopher: Schatten-IT; In: HMD – Praxis der Wirtschaftsinformatik 49 (2012), 288, S. 60-68."

Die gleiche Referenz in von Google Scholar bereitgestellten Stilen:

MLA Zimmermann, Stephan, and Christopher Rentrop. SSchatten-IT."HMD Praxis der Wirtschaftsinformatik 49.6 (2012): 60-68.

APA Zimmermann, S., & Rentrop, C. (2012). Schatten-IT. HMD Praxis der Wirtschaftsinformatik, 49(6), 60-68.

ISO690 ZIMMERMANN, Stephan; RENTROP, Christopher. Schatten-IT. HMD Praxis der Wirtschaftsinformatik, 2012, 49. Jg., Nr. 6, S. 60-68.

Abbildung 8. Anzeige Publikationsdaten in Wordpress.

Plugins für CMS wie Wordpress oder Typo3 strukturieren Publikationen oftmals in Tabellen. Der Text selbst folgt aber zumindest rudimentär gängigen Stilen.

Der verwendete Zitationsstil ist abhängig von mehreren Faktoren:

- technische Vorgaben (bspw. eingesetztes CMS)
- organisatorische Vorgaben (bspw. Rechtlinien in der Organisation)
- Studienfeld, so sind in der Informatik andere Stile gängig wie z. bsp. in anderen Naturwissenschaften oder den Geisteswissenschaften.
- Persönliche Vorlieben des Autors. Dies wirkt meist schwerer als alle anderern Vorgaben. Jeder Autor hat eine eigene Vorstellung davon, wie seine Referenzliste aussehen soll.

Unabhängig vom verwendeten Stil lässt sich jedoch prinzipiell beobachten, dass eine Publikation mindestens einen Autor hat, einen Titel besitzt, und zumindest ein Publikationsjahr trägt. Auch sind bei gängigen Stilen die einzelnen Felder irgendwie voneinander getrennt, z.b. mit ',' oder ';'. Betrachtet man o.g. Zitationsstile, so lassen sich Regeln ableiten, denen eine Referenz folgen muss. Die in Listing 7 gelisteten regulären Ausdrücke matchen jeweils die o.g. Zitationsstile. In Abhängigkeit von gefundenen Stilen auf identifizierten Publikationsseiten können diese relativ einfach um weitere Stile erweitert werden.

```
htwg = r'\w+,\s*\w+.*:\s*[a-zA-Z0-9_\s-]*;.*\(\d{4}\).*'
mla = r'\w+,\s*\w+.*."[a-zA-Z0-9_\s-]*\."\s*\w+.*\(\d{4}\)
    .*'
apa = r'\w+,\s*\w\..*\(\d{4}\)\.\s*[a-zA-Z0-9_\s-]*\.\s
    *.*,.*'
iso = r'\w+,\s*\w+.*\.\s*[a-zA-Z0-9_\s-]*\..*,\s*\d{4},'
```

Listing 7. Reguläre Ausdrücke zum Matchen von Publikationsstilen.

Der Code in Listing 8 testet, ob ein Publikationskandidat eine Publikation gemäß den vorher definitieren Zitationsstilen ist.

```
def isCitation(s):
        ret = False
        for style in citationstyles:
                ret |= (style.findall(s) != [])
                if ret == True:
                        break
        return ret
```

Listing 8. Identifiziere erlaubte Zitationen.

Kandidaten, die als Zitation identifiziert sind, da sie einem bekanntem Stil entsprechen, werden im folgenden Schritt geparst und strukturiert.

2.3 Strukturieren gefundener Einträge

Das Strukturieren gefundener Einträge ist beliebig aufwändig. Aufgabe ist, die extrahierten Publikationskandidaten zu untersuchen und Meta-Informationen wie Autoren, Titel, Buch, Jahr, etc. automatisch zu identifizieren. Dies ist ein als *citation parsing* bekanntes Problem. Einige Projekte und auch Web-Services versuchen bereits dieses Problem zu lösen. Allerdings beinhaltet *citation parsing* auch das parsen bereits strukturierter bibliographischer Einträge und deren Darstellung im Speicher. Da dies u.U. enorme linguistische Analysen erfordert, und aufgrund der hohen Anzahl unterschiedlicher Formate beliebig Komplex sein kann, sollte auf eine eigene und proprietäre Implementierung verzichtet und auf bereits existierende Mechanismen zurückgegriffen werden.

Es existieren einige öffentliche Suchmaschinen, die bereits die Suche nach bibliographischen Einträgen erlauben. Solche Portale suchen entweder in eigenen Datenbanken (wie WorldCat oder auch Pubmed) oder sie durchsuchen auch andere Datenbanken (Meta-Suchmaschinen wie Google Scholar oder auch Microsoft Academic Search). Der Such-String wird von solchen Suchmaschinen verwendet, um die Datenbanken nach passenden Einträgen zu durchsuchen. Gefundene Einträge werden dem Nutzer zurück gegeben. Oftmals gibt es zusätzlich die Möglichkeit Ergebnisse in unterschiedliche bibliographische Formate (wie bibtex) zu exportieren, woraufhin man direkt strukturierte Einträge zur Weiterverarbeitung erhält. Es gibt mehrere öffentliche bibliographische Suchmaschinen, die sich in wichtigen Punkten wie Ergebnisqualität, Nutzungsbedingungen, Exportmöglichkeiten, Vorhandensein und Mächtigkeit einer API, unterscheiden.

Der entscheidende Nachteil solcher bibliographischen Suchmaschinen ist, dass sie auf indexierte Einträge in bibliographischen Datenbanken angewiesen sind. Während diese Suchmaschinen sehr gute Ergebnisse produzieren können, sofern es passende Datenbankeinträge gibt, liefern sie leider gar kein Ergebnis für Suchen, die keine Entsprechung in Datenbanken haben. Sie dienen also nicht dazu, Publikationskandidaten zu strukturieren, die nicht bereits in einer Datenbank vorhanden sind. Auch ist man bei webbasierten Suchmaschinen an Nutzungsbedingen gebunden und dadurch in der Nutzung eingschränkt.

Mögliche bibliographische Suchmaschinen beinhalten die folgenden.

- Google Scholar
- Microsoft Academic Search
- Pubmed
- Crossref
- WorldCat

Neben den bibliographischen Suchmaschinen gibt es noch Software Projekte die auch ohne bibliographische Datenbanken versuchen Referenzen zu parsen und zu strukturieren. Hierbei gibt man einen Referenz-String an, dieser wird geparst, analysiert und strukturiert. Diese Projekte versuchen Meta-Informationen anhand der Referenz direkt zu extrahieren. Der Vorteil einer solchen Lösung ist, dass man auf keine Publikationsdatenbank angewiesen ist und somit auch Publikationskandidaten strukturiert werden können, die in keiner Datenbank indexiert sind. Da man sich das Suchen in Datenbanken erspart ist diese Lösung auch erheblich ressourcenschonender und um einiges schneller. Allerdings können die Ergebnisse u.U. ungenauer sein, da kein Abgleich mit bereits bekannten Einträgen erfolgt.

Quelloffene Projekte, die Meta-Daten-Extraktion aus Referenzen ermöglichen beinhalten die folgenden.

- FreeCite
- ParsCit
- Citeseer

Im folgenden werden die vorgestellten Portale und Software-Projekte auf ihre Anwendbarkeit untersucht.

Google Scholar Leider bietet Google bisher keine API für Scholar an [1]. Es gibt aber Open Source Parser wie scholar.py. Dieses bietet eine Reihe nützlicher Optionen mit denen nach Titeln, Autoren oder beliebigen Strings gesucht werden kann. Außerdem beherrscht es eine ganze Reihe verschiedener Ausgabeformate, u.a. Bibtex. Listing 9 zeigt ein Beispiel anhand des in Section 2.2 gefundenen Publikationseintrags.

```
python scholar.py -A "Zimmermann, Stephan; Rentrop,
    Christopher: Schatten-IT; In: HMD - Praxis der
    Wirtschaftsinformatik 49 (2012), 288, S. 60-68." --
    citation "bt"

@article{zimmermann2012schatten,
        title={Schatten-IT},
        author={Zimmermann, Stephan and Rentrop,
            Christopher},
        journal={HMD Praxis der Wirtschaftsinformatik},
        volume={49},
        number={6},
        pages={60--68},
        year={2012},
        publisher={Springer}
}

@article{wiener2013hmd,
        title={HMD Best Paper Award 2012},
        author={Wiener, Martin and Denk, Reinhard},
        journal={HMD Praxis der Wirtschaftsinformatik},
        volume={50},
        number={2},
        pages={110--113},
        year={2013},
        publisher={Springer}
}
```

Listing 9. Parsen und Konvertieren von Zitationen mit scholar.py.

Die ''Terms of Use'' schließen die Verwendung von Robots aus und Google scholar setzt nach einer bisher noch nicht genau identifizerter Anzahl an Anfragen captchas zum Schutz vor Robots ein, was die Anwendbarkeit drastisch reduziert. Es ist daher notwendig, Anfragen an Google weniger aggresiv vorzunehmen, d.h. mit reduzierter Frequenz, um einen längerfristigen Ausschluss zu vermeiden. Leider gibt es kein Nutzungsmodell seitens Google, das einen Einsatz im Projekt erlauben würde, auch das Fehlen einer geeigneten API spricht dagegen.

Microsoft Academic Search (MAS) bietet eine API [2] mit SOAP und JSON an. Um diese zu Verwenden muss man sich jedoch persönlich per E-Mail registrieren und die ‘’Terms of Use‘‘ akzeptieren.

Rechtlich spricht nichts gegen die Nutzung von MAS. Allerdings schreiben auch hier die ‘’Terms of Use‘‘ eine “verantwortungsvolle Nutzung” vor und Microsoft behält sich das Recht vor ohne Angabe von Gründen die Nutzung einzuschränken, was wieder auf das Vorhandensein eines Robot-Schutzes hindeutet.

Bei Tests hat sich ergeben, dass die Such-Ergebnisse auf MAS deutlich schlechter sind als bei Google Scholar. Die Suche nach o.g. Referenz liefert bspw. kein Ergebnis.

PubMed ist eine Publikationsdatenbank mit dem Fokus auf Medizinische und Life Science Veröffentlichungen. Auch PubMed bietet eine OAI-PMH Schnittstelle [3]. Da PubMed sehr spezialisiert ist, und im Vorfeld nicht automatisiert herausgefunden werden kann, in welchem Fachbereich eine Veröffentlichung angesiedelt ist, ist die Nutzung von PubMed nur bedingt sinnvoll. Wenn allerdings im Vorfeld die Art der Publikationen festehen, bspw. weil es sich um Publikationen einer Klinik o.ä. handelt, so lohnt sich PubMed evtl. mehr, als andere nicht spezialisierte Datenbanken.

CrossRef CrossRef ist eine Suchmaschine speziell für Publikations-Metadaten. Daher bietet sich der Einsatz hier an. CrossRef hat sich spezialisiert auf das Indexieren von *Document Object Identifiers*, also persistenter Identifikationsnummber von Dokumenten. CrossRefs Datenbank wird direkt gespeist durch teilnehmende Organisationen. Auch CrossRef bietet den Export strukturierter bibliographischer Daten in verschiedenen Formaten wie bibtex oder RIS an.

CrossRef bietet eine einfache API mittels formatierter GET-Requests an.

```
url = "http://search.crossref.org/?q=" + string
html = urlopen(url).read()
```

Listing 10. GET requests an CrossRef.

Auch gibt es open source Projekte, wie crossRefQuery die bereits das Exportieren formatierter bibliographischer Einträge mittels einer Suche auf CrossRef implementieren. Eine Suche bei CrossRef nach obiger Referenz liefert das folgende Ergebnis:

```
@article{Zimmermann_2012,
title={Schatten-IT},
volume={49},
ISSN={2198-2775},
url={http://dx.doi.org/10.1007/bf03340758},
DOI={10.1007/bf03340758},
number={6},
journal={HMD},
publisher={Springer Fachmedien Wiesbaden GmbH},
author={Zimmermann, Stephan and Rentrop, Christopher},
year={2012},
month={Dec},
pages={60--68}
}
```

Listing 11. Beispielergebnis einer CrossRef-Suche.

Crossref kann zuverlässig Referenzen finden, die auf Publikationen zeigen, die bei teilnehmenden Partnern veröffentlicht wurden. Während eine Webbasierte Suche den Export von bibliographischen Einträgen bietet, erlaubt die API leider nur einfache automatisierte Suchen nach DOIs. Es ist daher nicht unerheblich aufwändig mittels Crossref strukturierte Metadaten zu gewinnen.

WorldCat ist nach eigenen Angaben das größte Netzwerk von Bibliothek-Inhalten. Als solches ist es allerdings spezialisiert auf die Suche nach Beständen in Bibliotheken, d.h. Bücher und Audio/Video Medien. Auch lässt sich gezielt in bestimmten Bibliotheken suchen. WorldCat bietet eine Such-API an. Zur Suche bibliographischer Einträge wissenschaftlicher Veröffentlichungen eignet sich WorldCat auf Grund der Spezifizierung auf Bibliotheken nicht. So liefert eine Suche nach o.g. Referenz kein Ergebnis.

```
No results match your search for 'kw:Zimmermann, Stephan;
    Rentrop, Christopher: Schatten-IT; In: HMD - Praxis der
    Wirtschaftsinformatik 49 (2012), 288, S. 60-68.'.
```

Listing 12. Ergebnis einer Suche bei WorldCat.

ParsCit ParsCit ist ein quelloffenes Projekt der 'National University of Singapore', welches zwei Aufgaben erfüllt. 1) Parsen von Referenzen (*citation parsing* und 2) Parsen von wissenschaftlichen Dokumenten / Veröffentlichungen. Der Quellcode von ParsCit kann kostenfrei heruntergeladen und verwendet werden. Es ist aber auch möglich das Webportal oder einen Web Service zu verwenden. Hierfür stellen die Autoren bereits Perl Code und Beispiele bereit, die sehr einfach in bestehende Anwendungen integriert werden können.

ParsCit erkennt eine Vielzahl von Referenzstilen und liefert relativ gute Ergebnisse, die in den Formaten bibtex oder XML ausgegeben werden können.

```
@InCollection{d1e5,
author="Zimmermann, Stephan",
title="Rentrop, Christopher: Schatten-IT; In:",
booktitle="HMD -- Praxis der Wirtschaftsinformatik",
year="2012",
pages="288--60"
}

<citation>
<authors>
<author>Stephan Zimmermann</author>
</authors>
<booktitle>HMD - Praxis der Wirtschaftsinformatik</
    booktitle>
<volume>49</volume>
<date>2012</date>
<title>Rentrop, Christopher: Schatten-IT; In:</title>
<pages>288--60</pages>
```

```
</citation>
```

Listing 13. Ausgabeformate von ParsCit.

ParsCit wird auch verwendet von CiteSeerX (siehe unten), um Referenzen und wissenschaftliche Dokumente zu parsen.

FreeCite ist - ähnlich zu ParsCit und von diesem inspiriert - eine quelloffene Web Applikation, spezialisiert sich jedoch ausschließlich auf das Parsen von Referenzen. Wie ParsCit auch, kann der Quellcode heruntergeldaen und die Applikation selbst gehostet werden, oder die Web Applikation sowie ein Web Service der Autoren verwendet werden. Die API von FreeCite ist sehr einfach, selbst mit gängigen Kommandozeilentools verwendbar und leifert als Ergebnis strukturierte XML Dokumente. Die Ergebnisse selbst sind teilweise besser als bei ParsCit, wie folgendes Beispiel zeigt.

```
curl -H 'Accept: text/xml' -d "citation=Zimmermann, Stephan
    ; Rentrop, Christopher: Schatten-IT; In: HMD - Praxis
    der Wirtschaftsinformatik 49 (2012), 288, S. 60-68."
    http://freecite.library.brown.edu/citations/create

<citations>
        <citation valid='true'>
                <authors>
                        <author>Zimmermann Stephan</author>
                        <author>Rentrop Christopher</author
                            >
                </authors>
                <title>Schatten-IT; In: HMD - Praxis der
                    Wirtschaftsinformatik 49</title>
                <pages>60--68</pages>
                <year>2012</year>
                <raw_string>Zimmermann, Stephan; Rentrop,
                    Christopher: Schatten-IT; In: HMD -
                    Praxis der Wirtschaftsinformatik 49
                    (2012), 288, S. 60-68.</raw_string>
        </citation>
</citations>
```

Listing 14. Ergebnis einer Suche mit FreeCite.

CiteSeerX ist eine aktiv entwickelte digitale Bibliothek und Suchmaschine, die sich auf Veröffentlichungen in den Bereichen Informatik und Informationswissenschaften spezialisiert hat. CiteSeerX bietet eine Vielzahl an Funktionen, angefangen beim automatischen Harvesten von wissenschaftlichen Publikationen im Web, über Textextraktion und Parsen von Publikationen und Referenzen bis hin zum automatischen Indexieren und Archivieren von Publikationen und deren Metadaten.

CiteSeerX verwendet einen sog. fokussierten Crawler, um wissenschaftliche Publikationen, in Form von PDF und Postscript Dokumenten, von Webseiten zu Harvesten. Aus diesen wird der Text extrahiert und geparst. Dies geschieht

mit Hilfe von o.g. ParsCit. Aus dem geparsten Text werden die Dokument-Metadaten und die Referenzen extrahiert und strukturiert. Metadaten und die dazugehörigen Dokumente werden dann automatisch Indexiert und in einer Datenbank gespeichert. Das Web-Frontend dient als Suchmaschinen zur so aufgebauten digitalen Bibliothek.

CiteSeerX selbst bietet scheinbar keine API an. Obwohl es Publikationen bez. einer Citeseer API gibt [4] ist diese nicht öffentlich erreichbar. Allerdings gibt es eine OAI PMH Schnittstelle [5]. Es ist prinzipiell möglich Anfragen an CiteSeerX mit Hilfe des Referenz-Managers Jabref zu stellen. Dieser Verwendet ebenfalls CiteSeerX, um Metadaten zu erfragen. Laut Erfahrungsberichten [6] gibt es allerdings Probleme mit Such-Strings, die ":ënthalten.

Alternativ kann mittels geeigneter Scriptsprachen (bspw. python) eine Anfrage gestellt und die Ergebnisse geparst werden. Hierfür sendet man einen speziell formatierten GET Request. Die Ergebnis-Seite kann dann geparsed werden.

```
url = "http://citeseerx.ist.psu.edu/search?q="
q = "+" + string.replace(" ", "+") + "&submit=Search&sort=
    rlv&t=doc"
html = urlopen(url).read()
parsed = parse(html)
```

Listing 15. Suchanfragen an CiteSeerX.

Interessant ist jedoch weniger die angebotene Suchmaschine, als vielmehr das Software-Projekt CiteSeerX selbst. Denn CiteSeerX ist komplett quelloffen und kann selbst gehostet werden. Somit kann eine Institution seine eigene CiteSeerX-Instanz aufbauen und automatisch eine digitale Bibliothek durch Harvesten der eigenen Seiten generieren. Allerdings hat CiteSeerX relativ hohe Hard- und Softwareanforderungen, da es sich um eine komplexe und Komplette Lösung mehrerer Komponenten, wie verschiedene Harvester, Datenbank-, Web- und Applikationsserver handelt. CiteSeerX-Instanzen könnten als Service für andere Organisationen gehostet werden.

CiteSeerX konzentriert sich allerdings auf das Harvesten von wissenschafltichen Puklikationen und die Extraktion von Metadaten aus den Publikationen selbst. Der Crawler von CiteSeerX kann nicht nach Publikationslisten auf Autorenseiten suchen. Eine Integration einer solchen Funktionalität mit o.g. Techniken ist allerdings denkbar.

3 Related Work

Es gibt im Bereich der Bibliographie-Verwaltung bereits einige Bemühungen unstrukturierte Bibliographie-Einträge automatisch zu strukturieren und zu deduplizieren.

JabRef ist ein Referenz-Manager speziell für Bibtex. Es bietet eine Schnittstelle zu Citeseer, über die es möglich ist, Felder der bibliographischen Einträge von Citeseer herunter zu laden. Hierfür muss man allderdings die CiteSeerURL des Eintrags im vornherein kennen. JabRef kann auch mehrere Bibtex-Dateien zusammenführen und berichtet über Duplikate.

ReadCube ist - ähnlich wie Mendeley - ein Cloud-basierter Referenz-Manager. Er bietet Verbindungen zu mehreren Bibliographie-Datenbanken an (MAS, Google Scholar, CrossRef,PubMed, etc) um Publikationen zu Suchen, die Datenbank

abzugleichen, automatisch beim Import von Publikationen Metadaten zu extrahieren, und mehr.

Mendeley selbst bietet keine Schnittstelle zu Publikations-Datenbanken. Als Ersatz liefern sie allerdings ein Bookmarklet "Save to Mendeley" mit welchem auf Webseiten angezeigte Bibliographien direkt importiert werden können.

biblio-py ist ein in python geschriebener Bibliographie-Manager mit Such-Schnittstelle zur http://adsabs.harvard.edu/physics_service.html.

https://pypi.python.org/pypi/biblio.webquery/0.4.3b und das darauf aufsetzende [librarian](https://github.com/EvansMike/librarian) erlauben Web-basierte Suche nach Büchern anhand von ISBN / ISSN.

4 Bewertung und Ergebnis

Betrachtet man lediglich die Extraktion von Publikations-Metadaten aus unstrukturierten webbasierten Referenzlisten, so bietet sich der Einsatz eines gewöhnlichen Harvesters und Integration einer einfachen Heuristik zum Erkennen von Publikationslisten an. Die Extraktion von Referenzkandidaten kann über einfache reguläre Ausdrücke realisiert werden. Das Parsen und strukturieren erfolgt über den Web Service oder eine eigene Instanz von FreeCite, welches die einfachste API besitzt und sehr gute Ergebnisse liefert.

Sollen zusätzlich Metadaten aus Publikationen selbst extrahiert werden und ggf. sogar automatisch eine digitale Bibliothek aufgebaut werden, so liefert CiteSeerX fast alles, was dazu benötigt wird. Der Crawler muss jedoch um die Funktionalität zum erkennen von Publikationslisten erweitert werden. Des weiteren muss die Extraktion von Referenzkandidaten integriert werden. Die Strukturierung kann mit Hilfe des mitgelieferten ParsCit erfolgen.

Andere Lösungen kommen auf Grund von Einschränkungen in Funktionalität, Nutzungsrichtlinien, oder der Komplexität nicht in Frage.

Literatur

[1] Google, "Towards A Google Scholar API," online. [Online]. Available: http://wowter.net/2014/02/26/towards-google-scholar-api/ 2.3

[2] Microsoft, "Microsoft Academic Search API," online. [Online]. Available: http://academic.research.microsoft.com/about/ 2.3

[3] U. N. L. of Medicine, "[oai-pmh service." 2.3

[4] Y. Petinot, C. L. Giles, V. Bhatnagar, P. B. Teregowda, H. Han, and I. Councill, "CiteSeer-API: Towards Seamless Resource Location and Interlinking for Digital Libraries," in *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, ser. CIKM '04. New York, NY, USA: ACM, 2004, pp. 553–561. [Online]. Available: http://doi.acm.org/10.1145/1031171.1031275 2.3

[5] CiteSeerX, "Oai-pmh interface," online. [Online]. Available: http://csxstatic.ist.psu.edu/about/data 2.3

[6] Various, "CiteSeerX Search API," online. [Online]. Available: http://stackoverflow.com/questions/14085383/citeseerx-search-api 2.3