Analyse des Open-Access-Anteils bei Zeitschriftenartikeln: Anleitung und Dokumentation eines Python-Skripts

Eva Bunge[i], Michaela Voigt[ii]

Stand: 4. März 2018

Inhaltsverzeichnis

[i] Deutsches Museum, Bibliothek, ORCiD: 0000-0002-5587-5934
[ii] TU Berlin, Universitätsbibliothek, ORCiD: 0000-0001-9486-3189

1 Hintergrund

In Vorbereitung auf die Antragstellung bei der Deutschen Forschungsgemeinschaft (DFG) zur Förderung eines Open-Access-Publikationsfonds wurde an der Technischen Universität Berlin (TU Berlin) ein Verfahren für die Analyse des Publikationsaufkommen und den Anteil an Open Access (OA) entwickelt. Dieses Verfahren wurde weiterentwickelt für die Analyse des OA-Anteils wissenschaftlicher Zeitschriftenartikel der Angehörigen von Berliner Bildungs- und Forschungseinrichtungen.

Für die DFG-Antragsstellung benötigt wurden Aussagen über das Aufkommen von Zeitschriftenartikeln von TU-Angehörigen für die Jahre 2014 und 2015, insbesondere über den Anteil von Artikeln in Open-Access-Zeitschriften (Anteil OA Gold). Für die Analyse des Berliner Publikationsaufkommens von neun verschiedenen Einrichtungen wurden die Jahre 2013 bis 2015 analysiert. Für die Analyse der Berliner Artikeldaten für das Jahr 2016 wurde das Skript vollständig überarbeitet (insbesondere hinsichtlich Datenimport) sowie funktional erweitert (Anteil OA Grün).

Für die Analyse wurde auf Daten aus zehn bzw. sechzehn externen Literatur- und Zitationsdatenbanken zurückgegriffen. Die gewonnenen Daten zu den Dokumententypen `Article` bzw. `Review` wurden normalisiert, aggregiert und auf Dubletten geprüft. Um Artikel aus Open-Access-Zeitschriften zu identifizieren, wurden Daten des Directory of Open Access Journals (DOAJ)[1] genutzt. In den verbleibenden Daten von OA-Artikeln wurden im Folgenden diejenigen Artikel identifiziert, für die Angehörige der untersuchten Einrichtungen als Erst- oder Korrespondenzautoren angegeben sind. Um OA-Artikel in Closed-Access-Zeitschriften sowie den Anteil an Artikeln, die über den Grünen Weg Open Access verfügbar gemacht werden, zu identifizieren, wird die Schnittstelle von Unpaywall[2] abgefragt.

Bei der Datenerhebung wurden folgende Datenbanken berücksichtigt: Web of Science Core Collection, SciFinder (CAPlus), PubMed, TEMA, Inspec, IEEE Xplore, ProQuest Social Sciences, Business Source Complete, GeoRef, CAB Abstracts, CINAHL, Academic Search Premier, Embase, LISA, Scopus, Sport Discus.

Zur Unterstützung der Evaluation wurde ein Python-Skript entwickelt, dessen Funktionsweise ab S. 4 beschrieben wird. Ab S. 13 wird erläutert, welche Schritte für eine eigene Analyse mithilfe dieses Skripts durchzuführen sind.

Ob mit dem hier beschriebenen Verfahren alle OA-Artikel einer bestimmten Einrichtung identifiziert werden können, bleibt offen. Folgende Faktoren stellen potentielle Fehlerquellen dar:

- Artikel in Open-Access-Zeitschriften, die nicht in einer der geprüften Datenbanken indexiert sind, werden nicht berücksichtigt.
- Die Artikel werden über externe Datenbanken ermittelt; Voraussetzung für die Identifizierung ist das Erfassen der Affiliation in diesen Datenbanken. Es werden hier zwar Affiliationen für alle Autorinnen und Autoren erfasst – allerdings pro Autor bzw. Autorin

[1] Directory of Open Access Journals (DOAJ) s. http://doaj.org
[2] API von Unpaywall s. https://unpaywall.org/api/v2

meist nur eine Affiliation. Bei Mehrfachaffiliationen wird i. d. R. nur eine Institution in der Datenbank erfasst.

- Open-Access-Zeitschriften werden mithilfe des DOAJ identifiziert. Ist eine Zeitschrift nicht im DOAJ erfasst, werden Open-Access-Artikel nicht als solche erkannt. Es ist ebenso möglich, dass die Zeitschrift zum Zeitpunkt der Publikation des Artikels noch als Open-Access-Zeitschrift im DOAJ gelistet wurde, zum Zeitpunkt der Analyse aber aus dem DOAJ entfernt wurde.
- Während in einigen Datenbanken, z.B. Web of Science (WoS) oder Scopus, die Korrespondenzautorin bzw. der Korrespondenzautor gesondert ausgewiesen wird[3], werden in anderen Datenbanken lediglich Affiliationen erfasst. Ein eindeutiger (automatisch gestützter) Rückschluss auf die Korrespondenzautorschaft ist für diese Daten nicht möglich. Es werden für diese Datenbanken daher lediglich die Institutionsangaben zu Erstautorinnen bzw. Erstautoren evaluiert. Nicht in allen Disziplinen aber sind Korrespondenz- und Erstautorin identisch. Open-Access-Artikel, für die Angehörige einer Einrichtung Korrespondenz- nicht aber Erstautoren sind, bleiben somit unentdeckt.

  Eine Analyse der Daten des Jahres 2016 hat ergeben, dass in 14 % der Fälle keine explizite Angabe zur Korrespondenzautorschaft vorhanden war, woraufhin in erster Approximation der Erstautor bzw. die Erstautorin herangezogen wurde. Die Auswertung der Daten aus WoS zeigt, dass dies in ca. 64 % der Fälle korrekt ist. Insgesamt ergibt sich also eine daraus resultierende Fehlerquote von ca. 5 %.

[3] WOS: *reprint author*, Scopus: RIS-Feld N1

2 Funktionsweise des Python-Skripts

2.1 Aufbau und einzelne Funktionen

Das Skript ist in Python Version 2.7 geschrieben (zu Systemanforderungen vgl. Kap. 3.1 auf S. 13) und besteht aus dem Skript (`main.py`) sowie einer Hilfsdatei (`RIS-fields.csv`). In der Hilfsdatei ist für diejenigen Datenbanken, deren Daten im RIS-Format verarbeitet werden, definiert, welche RIS-Felder eingelesen werden sollen. Das Skript selbst ist in zwölf Teile gegliedert:

1. Funktionalitäten (de-) aktivieren Verschiedene Funktionalitäten des Skripts können aktiviert bzw. deaktiviert werden, zum Beispiel die Abfrage der Schnittstellen von Crossref und Unpaywall. Zur Erhöhung der Genauigkeit können Parameter für den gewünschten Untersuchungszeitraum festgelegt werden. Siehe die folgenden Punkte und Kap. 3.2 für eine Erläuterung der Funktionalitäten.

2. Klassen und Funktionen Es werden die Eigenschaften der untersuchten Institutionen, Datenbanken und Publikationen festgelegt und Funktionen definiert, die später mehrfach benötigt werden: der Dublettenabgleich, das Einlesen sowie die Normalisierung der verschiedenen Datenformate, die Abfragen der Schnittstellen von Crossref und Unpaywall sowie der Abgleich von Institutionsnamen.

3. Institutionen ansetzen Es wird festgelegt, welche Institutionen bei der Analyse berücksichtigt werden sollen. Es können beliebig viele Institutionen mit beliebig vielen Namensvarianten definiert werden. Im Skript sind exemplarisch mehrere Berliner Institutionen angelegt. Soll eine Institution nur kurzfristig von der Analyse ausgeschlossen werden, können alle Zeilen des jeweiligen Abschnitts ausgenommen werden, indem eine Raute (#) an den Zeilenanfang gesetzt wird. Soll eine Institution dauerhaft von der Analyse ausgeschlossen werden, sind die jeweiligen Zeilen zu löschen. Neue Institutionen können nach dem vorhandenen Muster angelegt werden (siehe auch Kap. 2.3 auf S. 8).

Für Institutionen mit eher generischem Namen wird eine zweite Option zur Institutionserkennung angeboten: Es können verschiedene Namensvarianten hinterlegt, für die das Skript eine exakte Suche (d. h. exakter Zeichenabgleich) in den Daten durchführt. Die exakte Suche kommt dann zum Tragen, wenn alle Autorinnen und Autoren (nicht nur Korrespondenzautor/in) auf Affilitionszugehörigkeit untersucht werden und damit eine teils sehr große Anzahl von Institutionennamen gleichzeitig ausgewertet wird.

4. Datenbanken ansetzen und Daten einlesen Es wird festgelegt, welche Datenbanken bei der Analyse verwendet werden. Jede Datenbank erhält einen Namen und eine Datenbank-ID.

Aus den im Vorfeld erhobenen Daten werden die relevanten Informationen eingelesen, extrahiert und einheitlich strukturiert. Jede Publikation erhält dabei die folgenden Eigenschaften: Autorinnen und Autoren, Titel, OA-Status, DOI, Zeitschrift, ISSN, eISSN, Verlag, Jahr, Affiliation, darin gefundene Namensvarianten, Korrespondenz- bzw. Erstautorschaft, darin gefundene Namensvariante, E-Mail-Adresse, Fachgebiet laut Datenbank[4], Angaben zu (Drittmittel-) Förderung[5], Fachgebiet laut DOAJ, Lizenz, Anmerkungen, die von Unpaywall gelieferten Daten zu diesem Artikel, Höhe der APC und deren Währung sowie die Datenbank, in der die Publikation gefunden wurde. Liegen keine Informationen zu einer Eigenschaft vor, so bleibt sie entweder leer oder es wird `None` als Wert eingetragen.

5. Dublettenabgleich Die Artikeldaten werden miteinander verglichen und Mehrfacheinträge eliminiert. Der Dublettenabgleich erfolgt in zwei Schritten: Es werden zunächst vorhandene DOIs verglichen. Sind keine DOIs vorhanden, werden Angaben zu Autorin bzw. Autor und Titel abgeglichen, indem die ersten drei Konsonanten der Autorennamen und die ersten 19 Konsonanten des Titels zu einem String zusammengefügt und dann verglichen werden. Die dazugehörigen Statistiken werden im Terminal ausgegeben.

Nachdem der Dublettenabgleich durchgeführt wurde, werden die generierten Daten in einer Datei `finalList` abgespeichert. Um bei einem nochmaligen Start des Skripts Zeit zu sparen, kann im Anschluss mithilfe der Variable `doReadIn` (s. Teil 1 des Skripts) bei Skriptstart die Datei `finalList` wieder eingelesen werden; nochmaliges Verarbeiten der Datenbankdaten und der Dublettenabgleich entfallen.

6. Erst- bzw. Korrespondenzautorschaft Es wird untersucht, welche Publikationen eine Autorin bzw. einen Autor der gesuchten Institution(en) als Erst- bzw. Korrespondenzautorin haben. Dazu werden die von den Datenbanken bereitgestellten Affiliationsangaben mit den in Abschnitt 3 des Skript definierten Namensvarianten der Institutionen abgeglichen. Zudem werden Autoren- und Affiliationsangaben auf die ersten 2500 Zeichen gekürzt. So werden Probleme durch sehr lange Autorenlisten bei der Weiterverarbeitung der Daten (z. B. in Tabellenkalkulationsprogrammen) vermieden.

Einige Datenbanken liefern keine bzw. uneinheitlich strukturierte Informationen zur Korrespondenzautorschaft, was eine automatisierte Verarbeitung erschwert bzw. unmöglich macht. Im nächsten Schritt werden daher die restlichen Publikationen in einer separaten Datei gespeichert (`docstobechecked.txt`). Es handelt sich erfahrungsgemäß um relativ wenige OA-Publikationen [6] , für die manuell die Affiliation der Korrespondenzautorin bzw. des Korrespondenzautors ermittelt werden muss. Die händisch ermittelten Daten können wieder in das Skript eingelesen werden (vgl. hierzu *10. Daten für manuelle Prüfung Korrespondenzautorschaft* auf S. 6).

[4] Zurzeit nur Web of Science und SciFinder

[5] Zurzeit nur Web of Science

[6] Für Analysen ausgehend von den im Skript angelegten Datenbanken lag der Anteil der Artikel bei unter einem Prozent. Werden bestimmte Datenbanken ausgenommen bzw. weitere hinzugefügt, kann sich der Anteil an Artikel verschieben, für die manuell die Affiliation der Korrespondenzautorin bzw. des Korrespondenzautors ermittelt werden muss.

7. DOAJ-Daten Die relevanten Informationen aus dem DOAJ werden eingelesen. Die ISSNs und eISSNs werden mit der Liste der gefundenen Publikationen abgeglichen. Die so gefundenen OA-Artikel in echten OA-Zeitschriften erhalten den OA-Status `gold` und werden mit Daten zu Fachgebiet, Verlag und Lizenz aus dem DOAJ angereichert. In der Ausgabedatei werden die so identifizierten Artikel mit dem Hinweis `Identified via DOAJ` in der Spalte `notes` markiert.

8. CrossRef-Daten Für Publikationen, die eine DOI aber keine ISSN haben, wird die Crossref-API[7] abgefragt. Die so ermittelten ISSNs werden erneut mit den DOAJ-Daten abgeglichen, um ggf. weitere Artikel in Open-Access-Zeitschriften zu identifizieren.

9. Unpaywall-Daten Für Publikationen, die eine DOI haben, wird die Unpaywall-API[8] abgefragt. So können weitere Open-Access-Artikel identifiziert werden. Als `hybrid` wird ein Artikel dann gewertet, wenn es laut Unpaywall eine OA-Version gibt, die über den Verlag direkt und unter einer freien Lizenz zugänglich ist. Als `green` wird ein Artikel dann gewertet, wenn es laut Unpaywall eine OA-Version gibt, die über ein gesichertes Repositorium zugänglich ist. In der Ausgabedatei werden die so identifizierten Artikel mit dem Hinweis `Identified via Unpaywall` in der Spalte `notes` markiert.

10. Daten für manuelle Prüfung Korrespondenzautorschaft Einige Datenbanken liefern keine bzw. uneinheitlich strukturierte Informationen zur Korrespondenzautorschaft. Um die Genauigkeit und Vollständigkeit der Analyse zu optimieren, können Artikel mit fehlenden Daten manuell überprüft werden. Hierzu sind die in der Datei `docstobechecked.txt` gelisteten Artikel zu prüfen. Die händisch ermittelten Daten können wieder in das Skript eingelesen werden, indem sie in Form einer tab-separierten Textdatei (ohne Kopfzeile) namens `docsChecked.txt` in der Codierung UTF-8 gespeichert und die Funktionalität zum Einlesen dieser Datei mithilfe der Variable `checkToDo` im ersten Teil des Skripts aktiviert wird. In der Datei `docsChecked.txt` muss jede Zeile einer Publikation entsprechen und in drei Spalten den Titel, die DOI und die gefundene Institution (wie er im Skript angesetzt ist) in dieser Reihenfolge enthalten. In der Ausgabedatei werden die so identifizierten Artikel mit dem Hinweis `Checked by hand` in der Spalte `notes` markiert.

11. APCs abschätzen und Ausgabedateien Im Terminal werden absolute Zahlen für die identifizierten OA-Artikel sowie erste Schätzwerte für APC-Kosten ausgegeben: Die Gesamtzahl von Artikeln in Open-Access-Zeitschriften, für die die Erst- bzw. Korrespondenzautorschaft bei

[7] Die Basis-URL für diese Abfrage lautet http://api.crossref.org/works/. Ein API-key ist für diese Abfragen nicht erforderlich. Eine ausführliche API-Dokumentation stellt Crossref online bereit, s. https://github.com/CrossRef/rest-api-doc/.

[8] Die Basis-URL für diese Abfrage lautet https://api.unpaywall.org/. Ein API-key ist für diese Abfragen nicht erforderlich, aber es muss in der Anfrage eine gültige E-Mail-Adresse angegeben werden. Eine API-Dokumentation stellt Unpaywall online bereit, s. https://unpaywall.org/api/v2.

der/den untersuchten Institution(en) liegt, wird mit 1481 €[9] multipliziert und die Resultate werden im Terminal ausgegeben. Sollen andere Durchschnittsgebühren zugrunde gelegt werden, kann im Skript der aktuelle Durchschnittswert ersetzt werden. Damit wird einen Überschlag über die anfallenden Kosten für Artikelgebühren pro Jahr errechnet.

Für eine fundierte Angabe zum Mittelbedarf sollten die Einzelartikel im Detail betrachtet werden, denn die Daten des OpenAPC-Projekts zeigen deutliche Schwankungen der durchschnittlichen Artikelgebühr: So hat die Technische Universität Dortmund im Schnitt 1031 €, die Universität Heidelberg im Schnitt 1480 € pro Artikel gezahlt.[10] Im Terminal werden daher zusätzlich Überschläge basierend auf Angaben zu APC-Kosten pro Zeitschrift laut DOAJ ausgegeben; diese Angabe erfolgt nach verschiedenen Währungen getrennt in absoluten Zahlen.

12. Statistik und Analyse Es werden die folgenden Auswertungen vorgenommen:

- Für jedes Jahr werden die Gesamtzahl a) der gefundenen Artikel sowie jeweils die Gesamtzahl der b) OA-Artikel in Open-Access-Zeitschriften, c) OA-Artikel in Hybridzeitschriften, d) OA-Artikel über den Grünen Weg und e) der OA-Artikel in Open-Access-Zeitschriften mit Erst- bzw. Korrespondenzautorschaft bei einer gesuchten Institution ausgegeben. Für alle Werte wird außerdem die Summe über alle untersuchten Jahre gebildet. Die Resultate werden in der Textdatei `statistics_OA.txt` gespeichert.
- Für OA-Artikel in OA-Zeitschriften (d. h. Artikel mit dem OA-Status `gold`) werden die vorhandenen Angaben zu Verlagen analysiert; es wird eine Statistik über die Häufigkeitsverteilung der OA-Artikel auf die vorhandenen Verlage erstellt. Die Ergebnisse der Analyse werden in der Datei `statistics_publishers.txt` gespeichert. Hier ist zu beachten, dass die Verlagsnamen nicht normiert sind und daher noch händisch bereinigt werden müssen.

2.2 Ausgabedateien für weitere Analyse

Es werden die folgenden Informationen in tab-separierte Textdateien geschrieben, die für detaillierte Auswertungen in Tabellenkalkulationsprogramme (bspw. Excel) importiert werden können:

- `allPubs.txt` = Liste aller gefundenen Artikel
- `docsToBeChecked.txt` = Liste der Publikationen, für die es nicht möglich ist, die Erst- bzw. Korrespondenzautorschaft automatisch zu ermitteln
- `oaDOI-response.txt` = Liste der Publikationen, für die die Unpaywall-API abgefragte wurde inkl. einzelne Datenfelder dieser Schnittstelle (OA-Status Artikel, OA-Status Journal, Quelle für OA-Version, freie Lizenz, Verlag, OA-Farbe)

[9] Es handelt sich dabei um die durchschnittliche Gebühr für Artikel in OA-Zeitschriften der an OpenAPC teilnehmenden Institutionen, vgl. https://github.com/OpenAPC/openapc-de (Stand 21. September 2017).

[10] Vgl. Zahlen zu Artikelgebühren der an OpenAPC-teilnehmenden Institutionen, https://github.com/OpenAPC/openapc-de (Stand 21. September 2017).

- `statistics_OA.txt` = Statistik der analysierten Artikel
- `statistics_goldPublishers.txt` = Häufigkeitsverteilung der Gold-Artikel (OA-Artikel in OA-Zeitschriften) auf Verlage
- `DOIs-oaDOI-error.txt` = Liste der DOIs, für die Unpaywall eine Fehlermeldung zurückgegeben hat

Die Datei `allPubs.txt` ist wie in Abschnitt *4. Datenbanken ansetzen und Daten einlesen* (vgl. S. 4) beschrieben formatiert. Eine Übersicht über alle Felder sowie deren Herkunft ist dem Anhang zu entnehmen (vgl. Tab. 5 auf S. 28).

2.3 Eine neue Institution ansetzen

Im Skript können beliebig viele Institutionen gleichzeitig verarbeitet werden. Für jede Institution können außerdem beliebig viel Namensvarianten angesetzt werden, nach denen in den Affiliationsangaben der Datenbanken gesucht wird. Neue Institutionen lassen sich wie folgt ansetzen:

In Abschnitt drei des Skripts die Institution nach dem vorhandenen Muster initialisieren:

```
# TU Berlin initialisieren
TUnames = [['Tech', 'Univ', 'Berlin'], ['Berlin', 'TU'],
           ['Berlin', 'Inst', 'Technol']]
TU = inst('TU', TUnames)

# HU Berlin initialisieren
HUnames = [['Humboldt', 'Univ', 'Berlin'], ['Berlin', 'HU']]
HU = inst('HU', HUnames)
```

Dabei entspricht der Ausdruck `[['Humboldt','Univ','Berlin'], ['Berlin','HU']]` der Suche `(('Humboldt' AND 'Univ' AND 'Berlin') OR ('Berlin' AND 'HU'))`.

Die obigen Namensvarianten sind möglichst allgemein formuliert – so deckt die Buchstabenkombination `Univ` sowohl die Begriffe `Universität` und `University`, als auch die Abkürzung `Univ` ab. Die allgemeine Ansetzung der Namensvarianten erleichtert so die Suche. Bei Institutionen wie der Humboldt-Universität zu Berlin, die einen relativ distinktiven Namen hat, verursacht dies keine großen Probleme. Bei Institutionen wie der Technischen Universität Berlin kann dies jedoch zu Problemen führen. Gehört eine Autorin bzw. ein Autor zum Beispiel der Technischen Universität Dresden und einer beliebigen anderen Institution in Berlin an, so würden die obigen Namensvarianten eine Zugehörigkeit zur TU Berlin verzeichnen. Um dieses Problem zu umgehen, kann im Skript eine zweite Liste mit Namensvarianten angelegt werden. Wird diese Option genutzt, sollten so viele verschiedene Varianten wie möglich angelegt werden, da im Skript dann lediglich ein genauer Zeichenabgleich erfolgt von den hier hinterlegten Varianten mit den vorhandenen Affiliationsangaben in den Datenbanken. Dies muss am Ende des dritten Abschnitts nach dem folgenden Muster geschehen:

```
TU.nameVar1 = [['Technische Universitat Berlin'],
               ['Technische Universitaet Berlin'],
               ['Technische Universität Berlin'],
               ['Berlin Institute of Techn'],
               ['Tech Univ Berlin'],
               ['Berlin Univ Technol'],
               ['Univ Technol Berlin'],
               ['TU Berlin'],
               ['Tech. Univ. Berlin'],
               ['Berlin Inst Technol'],
               ['Technical University Berlin'],
               ['Technische Universitaet de Berlin'],
               ['Technical University of Berlin'],
               ['Berlin University of Technology']]
```

Zurzeit werden im Skript die allgemeinen Namensvarianten für die Affiliationssuche in der Angabe der Korrespondenzautorschaft verwendet, da hier in vielen Fällen nur eine Affiliation angegeben ist. Die zweite Variante wird verwendet, wenn die Liste aller Affiliationen eines Artikels durchsucht wird.

Falls keine zweite Liste mit Namensvarianten angegeben wird, werden die ursprünglichen Namensvarianten verwendet. In dem Skript ist diese Liste mit Namensvarianten aktuell für die TU Berlin aktiv. Sie kann kurzfristig ausgenommen werden, indem eine Raute (#) an den Anfang jeder Zeile in diesem Abschnitt gesetzt wird. Um diese Option dauerhaft auszuschließen, sind die jeweiligen Zeilen im Skript zu löschen.

2.4 Eine Datenbank von der Analyse ausschließen

Jede Datenbank hat zwei Abschnitte im vierten Teil des Skripts. Der erste Abschnitt umfasst nur eine Zeile und definiert Namen und ID-Nummer der Datenbank. Im zweiten Abschnitt werden die Daten aus der Datenbank eingelesen. Web of Science ist ein elementarer Teil des Skripts und kann nicht von der Analyse ausgeschlossen werden. Alle anderen Datenbanken lassen sich wie folgt ausschließen:

1. Ersten Abschnitt finden. Dieser befindet sich im ersten Textblock des vierten Teils des Skripts und ist wie das folgende Beispiel formuliert:

   ```
   dbLisa = Database('LISA', 15)
   ```

2. Zweiten Abschnitt finden. Dieser befindet sich am Ende des vierten Teils des Skripts. Der Name der Datenbank wird explizit im zugehörigen Kommentar erwähnt, zum Beispiel:

   ```
   # Read in 'LISA' file and extract relevant information.
   dbLisa.content = risFormat('input-files/lisa2016.ris', dbLisa.idNummer)
   print 'Finished reading in LISA'
   ```

3. Falls die Datenbank permanent von der Analyse ausgeschlossen werden soll, können alle Zeilen dieser beiden Abschnitte gelöscht werden. Soll die Datenbank nur kurzzeitig ausgeschlossen werden, können alle Zeilen der beiden Abschnitte auskommentiert werden, indem jeweils eine Raute (#) an die Zeilenanfänge gesetzt wird.

2.5 Eine neue Datenbank aufnehmen

Eine Liste der aktuell integrierten Datenbanken ist Kap. 3.3 ab S. 16 zu entnehmen. Eine neue Datenbank kann einfach zur Auswertung hinzugefügt werden (Fall A), wenn die Daten mithilfe von Citavi in das WOS-Datenformat umgewandelt werden. Diese Methode wurde in einer früheren Version des Skripts z. B. für die Datenbanken TEMA und ProQuest verwendet. Eine häufig bessere Datenqualität wird beim direkten Einlesen von RIS-Daten erreicht; hierzu muss jedoch vorab eine detaillierte Analyse der RIS-Felder erfolgen und das entsprechende Mapping in der Hilfsdatei `RIS-fields.csv` eingetragen werden (Fall B).

Eine Datenbank lässt sich wie folgt hinzufügen:

1. Abfrage in der Datenbank durchführen und die Vorgehensweise dokumentieren.
2. Resultate exportieren (z. B. im RIS-Format).
3. Im vierten Teil des Skripts zwei neue Abschnitte für diese Datenbank hinzufügen:
   a) Namen, Datenbank-ID und Variablennamen für die Datenbank nach dem vorhandenen Muster am Anfang des vierten Teils des Skripts anlegen (im Textblock, der auf den Kommentar `# Set up databases` folgt). Für eine Datenbank namens *Testa* könnte dies zum Beispiel wie folgt aussehen:

      ```
      dbTesta = Database('Testa', 77)
      ```

      Dabei entspricht `dbTesta` dem Variablennamen, der intern im Skript für die Datenbank verwendet wird, `Testa` ist der Name der Datenbank und 77 ist die Datenbank-ID. Alle drei Werte sind beliebig, müssen aber im Skript eindeutig sein.

   b) Zeilen im Skript nach dem vorhandenen Muster im vierten Teil des Skripts ergänzen, um die Daten einlesen zu lassen. Für unsere Datenbank *Testa* würde dies für Fall A (WOS-Format) wie folgt aussehen:

      ```
      # Read in 'Testa' file and extract relevant information.
      dbTesta.content = wosFormat('input-files/testa20xx.txt',
      dbTesta.idNummer)
      print 'Finished reading in Testa'
      ```

   c) Definieren, in welchem Format die Daten eingelesen werden sollen (`wosformat` oder `risformat`). Für unsere Datenbank *Testa* würde dies für Fall B (RIS-Format) wie folgt aussehen:

      ```
      # Read in 'Testa' file and extract relevant information.
      dbTesta.content = risFormat('input-files/testa20xx.txt',
      ```

```
dbTesta.idNummer)
print 'Finished reading in Testa'
```

Vorsicht: Neue Datenbanken müssen vor dem letzten Abschnitt des vierten Skriptteils hinzugefügt werden, d. h. vor der Zeile `# Transform all characters in DOIs to lower case.`

4. Einlesen der Daten vorbereiten (Fall A oder B):

   A) Resultate in Citavi importieren, aus Citavi im WOS-Datenformat exportieren und im gleichen Verzeichnis wie sonstige Datenbankergebnisse ablegen (Dateiname z. B. `testa20xx.txt`).

   B) Neue Spalte für die neue Datenbank in der Hilfsdatei `RIS-fields.csv` anlegen[11] und dabei die im Python-Skript hinterlegte Datenbank-ID als Spaltennamen eintragen. Die aus der Datenbank exportierten Daten detailliert analysieren und in der CSV-Datei die passenden RIS-Felder zuordnen. Bei dieser Variante muss berücksichtigt werden, dass z. B. Datumsangaben in den unterschiedlichen Datenbanken nicht einheitlich formatiert werden. Die üblichsten Formate sind bereits berücksichtigt, hier liegt allerdings eine potentielle Fehlerquelle. Auch sind im RIS-Format Angaben zu Affiliationen und Korrespondenzautorschaft oft in einem Feld mit anderen Informationen angegeben, die damit das Ergebnis verfälschen können. In diesem Fall wäre es nötig, im zweiten Skriptteil die Funktion `risFormat` individuell anzupassen.

2.6 Potentielle Fehlerquellen

Die folgenden Faktoren können Fehler in der Analyse mithilfe des hier beschriebenen Skripts verursachen:

- In WoS werden alle Titel ins Englische übersetzt. Ist ein Titel in einer anderen Datenbank mit dem deutschen Titel verzeichnet und hat dieser Artikel keine DOI, würde diese Publikation doppelt gezählt.
- Ist der Autorenname sehr kurz bzw. enthält nur wenige Konsonanten (z. B. Lee, Zhi), kann es zu Fehlern beim Dublettenabgleich kommen, falls eine Datenbank nur den ersten Buchstaben des Vornamens, eine andere Datenbank aber den vollständigen Vornamen erfasst.
- Es kann zu Fehlern beim Dublettenabgleich kommen, wenn der Nachname der Autorin bzw. des Autors aus zwei Wörtern (z. B. da Silva) besteht und in den Datenbanken uneinheitlich nachgewiesen ist (da Silva, H. oder Silva, H. da).
- Zur Bestimmung des Anteils hybrider OA-Artikel werden bestimmte Annahmen getroffen (hybrid = OA-Version unter freier Lizenz über Verlag direkt zugänglich). Werden andere Annahmen getroffen (z. B. OA-Version über Verlag direkt zugänglich – auch ohne

[11] Als Feldtrennzeichen sind Semikolons zu nutzen.

freie Lizenz), werden durch das Skript ggf. nicht alle entsprechenden Artikel identifiziert. Hinweise zu manuellen Prüfung weiterer OA-Artikel sind 3.4 ab. S. 23 zu entnehmen.

- Bei Academic Search Premier handelt es sich um eine Meta-Datenbank, über die viele verschiedene Datenbanken gleichzeitig abgefragt werden können. Die produzierte Ausgabedatei enthält dadurch eine relativ hohe Anzahl an Dubletten sowie Publikationsdaten in vielen verschiedenen Metadatenformaten. Dadurch wird die Verarbeitung erschwert und Daten aus dieser Quelle sind besonders fehleranfällig.

2.7 Mögliche Erweiterungen

Bei der Erstellung des Skripts wurde auf ein möglichst effizientes Aufwand-Nutzen-Verhältnis geachtet. Daher wurden einige denkbare Erweiterungen und Verbesserungen des Skripts bislang nicht umgesetzt:

- Dublettencheck verbessern: Eine weitere denkbare Methode für den Dublettencheck besteht darin, aus ISSN, Jahrgang, Ausgabe und Seitenzahlen einen String zu bilden. So erhält man einen weiteren eindeutigen Identifikator, der zum Abgleich herangezogen werden kann.
- Ausschlusskriterien: Um die Institutionssuche in den Affiliationsangaben zu verbessern, könnte eine Liste von Wörtern festgelegt werden, die nicht in der Affiliationsangabe vorkommen dürfen. So besteht für die TU Berlin eine gewisse Verwechslungsgefahr mit der Beuth Hochschule für Technik Berlin, insbesondere wenn gleichzeitig der englische Name (Beuth University of Applied Sciences) angegeben wird. Um dieses Problem zu umgehen, könnte man alle Artikel, deren Affiliationsangaben das Wort `Beuth` enthalten, automatisch von der Zuordnung zur TU Berlin ausschließen.

3 Eine Analyse durchführen

3.1 Systemanforderungen

Das Skript wurde für die Python-Version 2.7 entwickelt. Aktuelle Python-Versionen können für verschiedene Betriebssystemplattformen (u. a. Windows, Mac OS X und Linux) frei heruntergeladen werden.[12] Aufgrund der im Skript eingebundenen Pakete sollte mindestens die Python-Version 2.7.11 installiert sein. Je nach lokal vorhandenem System sind ggf. einzelne Pakete nachzuinstallieren.

Bei Python handelt es sich um Open-Source-Software, die kostenfrei heruntergeladen werden kann. Die Python Software Foundation stellt online Hinweise zur Installation und Handhabung sowie eine ausführliche Dokumentation und ein FAQ bereit[13]. Allen, die noch nie ein Skript in einem Terminal oder einer Programmierumgebung gestartet haben, wird die Lektüre der Seite „Python for Non-Programmers“[14] empfohlen.

Während das Skript selbst auf verschiedenen Betriebssystemplattformen lauffähig ist, wird empfohlen, die Dateien für das Einlesen auf einem Windows-System vorzubereiten: Es wurden Probleme beim Einlesen von Dateien festgestellt, die auf einem Mac-System[15] vorbereitet wurden.

Das Skript kann Daten in zwei nativen Formaten, Web of Science und PubMed, sowie im RIS-Format einlesen. Zwar handelt es sich bei RIS um ein standardisiertes Format, aber der Standard bietet mitunter verschiedene Felder[16], lässt verschiedene Feldbelegungen zu[17] oder bietet Freitextfelder, in denen etwa Angaben zu Affiliationen oder Projektförderung hinterlegt werden können. Hinzu kommt, dass sich die Verwendung der RIS-Felder durch verschiedene Datenbankanbieter über die Zeit zu ändern scheint. Vor diesem Hintergrund wurde ein Mapping der gesuchten Angaben auf die jeweiligen RIS-Felder pro Datenbank erstellt (s. Hilfsdatei `RIS-fields.csv`), in der Änderungen bei Feldbelegungen eingetragen werden können.

Soll eine neue Datenbank im RIS-Format in die Analyse aufgenommen werden, sollte eine detaillierte Untersuchung der RIS-Daten vorangehen und die passenden Felder in dieser Datei eingetragen werden. Neue Datenbanken können dann einfacher in die Analyse aufgenommen werden, wenn die exportierten Daten vorab in das WOS-Format konvertiert wurden. Dies kann

[12] Download unter https://www.python.org/downloads/. Auf aktuellen Mac- und Linux-Systemen ist Python standardmäßig vorhanden; ggf. muss aber die Version upgedatet werden.

[13] Python-Dokumentation s. https://docs.python.org, Hinweise für AnfängerInnen s. insbes. https://www.python.org/about/gettingstarted/

[14] https://wiki.python.org/moin/BeginnersGuide/NonProgrammers

[15] Getestet wurde ein Zusammenführen von Dateien (Export aus verschiedenen Datenbanken) unter Mac OS X 10.9.5 und dem Texteditor TextWrangler 5.5.1. Bei dem Vorbereiten von Dateien in einem Mac-Terminal mithilfe von `cat` wurden bisher keine Probleme festgestellt. Eine Vorbereitung der Dateien in einer Linux-Umgebung wurde bisher nicht getestet.

[16] So kann der Titel der Zeitschrift z. B. in den Feldern `JO`, `JF` oder `T2` angegeben werden.

[17] Zwei Beispiele hierfür sind ISSNs und DOIs: ISSNs können z. B. in den Formen `1234-5678`, `12345678 (ISSN)` oder `12345678`, DOIs in den Formen `10.123/456789`, `http://dx.doi.org/10.123/456789` oder `https://doi.org/10.123/456789` angegeben sein.

mithilfe von Citavi erfolgen.[18] Dabei ist jedoch zu beachten, dass eventuell vorhandene Angaben zu Affiliationen bei einer Konvertierung verloren gehen.

Input-Dateien sollten in einem Unterorder `input-files` abgelegt werden, der im gleichen Ordner liegt wie das zu startende Python-Skript. Output-Dateien werden in einem Unterordner `output-files` abgespeichert. Daher müssen für diese Arbeitsordner sowohl Lese- als auch Schreibrechte vorhanden sein.

3.2 Handhabung des Skripts

Um den Open-Access-Anteil bei Zeitschriftenartikeln mithilfe des Skripts zu analysieren, müssen die im Folgenden beschriebenen Schritte durchgeführt werden. Dabei können Publikationen einer oder mehrerer Institutionen gleichzeitig analysiert werden. Eine gleichzeitige Analyse mehrerer Jahre ist ebenfalls möglich.

1. Abfrage in den gewünschten Datenbanken (Beschreibung vgl. S. 16 ff)
2. DOAJ-Daten herunterladen und vorbereiten (Beschreibung vgl. S. 23)
3. Im dritten Abschnitt des Skripts Namensvarianten für zu untersuchende Institutionen eintragen (Beschreibung vgl. S. 4).
4. Zwei neue Unterordner `input-files` und `output-files` in dem Verzeichnis anlegen, in dem das Skript `main.py` abgelegt ist.
5. Dateien mit Ergebnissen aus Datenbankrecherchen und DOAJ-Daten in dem Unterordner `input-files` ablegen.
6. Im ersten Abschnitt des Skripts gewünschte Funktionalitäten auswählen:
    - Datenauswertung (Statistik, Diagramme)
        - `doAnalysis = True`: ruft den letzten Abschnitt des Skripts für erste Datenauswertungen auf
        - `doAnalysis = False`: Funktionalität deaktivieren
    - Einlesen Artikeldaten
        - `doReadIn = True`: beim ersten Starten des Skripts wählen, liest Daten aus Datenbankrecherchen etc. ein
        - `doReadIn = False`: Funktionalität deaktivieren, um Daten aus Zwischenspeicher einzulesen (verkürzt Skriptlaufzeit)
    - Abfrage der Crossref-API
        - `contactCR = 0`: Funktionalität deaktivieren

[18] Erfolgreich getestet wurde dies mit Citavi5, Download unter https://www.citavi.com/de/download.html. Bei Citavi handelt es sich um eine lizenzpflichtige Software, für die viele deutsche Forschungseinrichtungen eine Campuslizenz erworben haben. Citavi kann nativ aktuell nur auf Windows-Betriebssystemen installiert werden. Um Citavi auch auf anderen Plattformen nutzen zu können, kann Windows in einer virtuellen Maschine installiert werden. Hinweise, wie Citavi auf dem Mac genutzt werden kann, hält das Citavi-Handbuch bereit: https://www.citavi.com/sub/manual5/de/index.html

  - `contactCR = 1`: beim ersten Start des Skripts wählen, kontaktiert Crossref-API und ergänzt fehlende ISSNs
  - `contactCR = 2`: für wiederholtes Starten des Skripts wählen, liest Ergebnisse aus dem Zwischenspeicher ein (verkürzt Skriptlaufzeit)
- Abfrage der Unpaywall-API
  - `contactOaDOI = 0`: Funktionalität deaktivieren
  - `contactOaDOI = 1`: beim ersten Start des Skripts wählen, kontaktiert Unpaywall-API und ermittelt OA-Artikel in Closed-Access-Zeitschriften (`hybrid`) bzw. frei zugängliche Versionen in Repositorien (`green`)
  - `contactOaDOI = 2`: für wiederholtes Starten des Skripts wählen, liest Ergebnisse aus dem Zwischenspeicher ein (verkürzt Skriptlaufzeit)
  - `eMail = $email`: für Abfrage der API (`contactOaDOI = 1`) eine gültige E-Mail-Adresse eintragen, über die der/die API-NutzerIn kontaktiert werden kann (Pflichtangabe)
- Manuelle Prüfung Korrespondenzautorschaft
  - `checkToDo = 0`: Funktionalität deaktivieren
  - `checkToDo = 1`: beim ersten Starten des Skripts wählen, erstellt Datei `docstobeChecked.txt`
  - `checkToDo = 2`: für wiederholtes Starten des Skripts wählen, liest Ergebnisse aus der manuell erstellten Datei `docsChecked.txt`
- Auswahl der relevanten Publikationsjahre (ist die Analyse für nur ein Jahr gewünscht, ist für beide Variablen das gleiche Jahr einzutragen)
  - `yearMin = $year`: Startjahr des gewünschten Untersuchungszeitraumes eintragen
  - `yearMax = $year`: Endjahr des gewünschten Untersuchungszeitraumes eintragen

7. Skript in Python-Umgebung[19] starten
   (`doAnalysis = True`, `doReadIn = True`, `contactCR = 1`, `checkToDo = 1`)

8. Die Ausgabedatei `docstobechecked.txt` nachbereiten: Für alle Einträge in der Datei prüfen, ob für den Artikel die Korrespondenz- bzw. Erstautorschaft bei einer Autorin bzw. einem Autor der untersuchten Einrichtung(en) liegt. Ergebnisse in neuer Datei mit Dateinamen `docsChecked.txt` speichern und im Ordner `input-files` ablegen. In der Datei `docsChecked.txt` muss jede Zeile einer Publikation entsprechen; die drei Spalten müssen den Titel, die DOI und die gefundene Namensvariante (in dieser Reihenfolge) enthalten.[20]

9. Skript in Python-Umgebung starten
   (`doAnalysis = True`, `doReadIn = False`, `contactCR = 2`, `checkToDo = 2`)

[19] Das Skript kann in einer Python-Programmierumgebung (IDE) (s. u. a. https://wiki.python.org/moin/IntegratedDevelopmentEnvironments) oder vom Terminal des Betriebssystems aus gestartet werden (s. u. a. https://en.wikibooks.org/wiki/Python_Programming/Creating_Python_Programs).

[20] Es ist eine tab-separierte Textdatei in UTF-8-Kodierung ohne Kopfzeile (d. h. ohne Zeile für Spaltennamen) anzulegen; für nähere Beschreibung vgl. S. 5.

10. Ausgabedateien auswerten: Für eine Beschreibung der Ausgabedateien vgl. Abschnitt *10. Statistik und Analyse* auf S. 7, für eine Übersicht der Dateien und Felder vgl. Anhang ab S. 26). Es wird empfohlen, die Daten vor der Auswertung zu bereinigen; entsprechende Empfehlungen finden sich in 3.4 auf S. 23.

3.3 Abfrage der Datenbanken und Aufbereitung der Daten

Im Folgenden wird jeweils ein Weg beschrieben, wie die Abfragen in den einzelnen Datenbanken durchgeführt werden können[21] und wie die Daten des DOAJ aufbereitet werden müssen.

Academic Search Premier

- Suchabfrage (einfache Suche):
  - Beispiel: `AF (tu berlin* OR tech* univ* berlin* OR technisch* universität berlin* OR berlin* inst* technol*)`
- Suchergebnisse filtern nach
  - Jahr
  - Dokumententyp: `publication type = Academic Journals`
- Daten exportieren:
  - `Share`
  - `Export results: E-mail a link to download exported results`
  - Format auswählen: `RIS format`
  - Formular aufüllen (E-Mail-Adresse)
  - `Send`
  - Link in E-Mail öffnen und Datei abspeichern
- Speichern: Dateiname `ebsco20xx.txt`

Business Source Complete

- Suchabfrage (einfache Suche) via EBSCOhost (Anführungszeichen nutzen für exakte Suche!):
  - Beispiel: `AD "tech* univ* berlin" OR AD "tu berlin" OR AD "berlin inst* tech*"`
- Suchergebnisse filtern nach
  - Jahr
  - Dokumententyp: `publication type = Academic Journals`
- Daten exportieren:
  - `Share`
  - `Add to Folder: Results`
  - `Folder`

[21] Beispielabfragen sind für ein Jahr formuliert, aber das Skript kann mehrere Jahrgänge gleichzeitig verarbeiten. Die Beispiele sind (mit Ausnahme von TEMA) dokumentiert für eine englischsprachige Oberfläche der jeweiligen Datenbanken.

  - Select all
  - Export
  - RIS format
  - Save
- Speichern: Dateiname bsc20xx.txt

CAB Abstracts

- Suchabfrage (erweiterte Suche) via OvidSP:
  - Beispiel: (tech* univ* berlin or TU* Berlin).in. OR (tu-berlin de).ma.
- Suchergebnisse filtern nach
  - Jahr
  - Dokumententyp: Publication Type = Journal Article
- Daten exportieren:
  - Range: 1-200
  - Export
  - Export To: RIS
  - Select Fields to Display: Complete Reference
  - Export Citation(s)
- Datenexport wiederholen für alle vorhandenen Datensätze
- alle Ergebnisse in *eine* Datei kopieren (Spaltennamen der Einzeldateien nur einmal und als erste Zeile übernehmen!)
- Speichern: Dateiname cab20xx.txt

Cumulative Index to Nursing & Allied Health Literature (CINAHL)

- Suchabfrage (erweiterte Suche) via EBSCOhost:
  - Beispiel: AF tech* univ* berlin OR AF TU Berlin OR AF berlin inst* technol*
- Suchergebnisse filtern nach
  - Jahr
  - Dokumententyp: Publication Type = Academic Journals
- Daten von MEDLINE von der Suche ausschließen
- Daten exportieren:
  - Share
  - Page Options: max. Treffermenge auswählen
  - Add to Folder: Results 1-50
  - Folder
  - Select all
  - Export
  - Direct Export in RIS format
  - Save
- Speichern: Dateiname cinahl20xx.txt

Embase

- Suchabfrage (Expertensuche) via OvidSP:
  - Beispiel: `keyword(tech* univ* berlin.ad. OR tech* univ* berlin.in. OR tech* univ* of berlin.ad. OR tech* univ* of berlin.in. OR TU Berlin.ad. OR TU Berlin.in. OR Berlin Inst* Technol*.ad. OR Berlin Inst* Technol*.in. OR Berlin Inst* of Technol*.ad. OR Berlin Inst* of Technol*.in.)`
- Suchergebnisse filtern nach
  - Jahr
  - Dokumententyp: `Publication Type = Article, Review` (hierzu Filter über `Additional Limits` bzw. `Zusätzliche Eingrenzungen` auswählen)
- Daten exportieren:
  - `Range: 1-200`
  - `Export`
  - `Export to: RIS`
  - `Select Fields to Display: Complete Reference`
  - `Export Citation(s)`
- Datenexport wiederholen für alle vorhandenen Datensätze (OvidSP erlaubt Herunterladen von max. 200 Datensätzen pro Durchgang – ggf. abhängig von Lizenz)
- alle Ergebnisse in *eine* Datei kopieren (Spaltennamen der Einzeldateien nur einmal und als erste Zeile übernehmen!)
- Speichern: Dateiname `embase20xx.txt`

GeoRef

- Suchabfrage (einfache Suche) via EBSCOhost:
  - Beispiel: `tech* univ* berlin (All text)`
- Suchergebnisse filtern nach
  - Jahr
  - Dokumententyp: `source type = Academic Journals`
- Daten exportieren:
  - `Share`
  - `Add to Folder: Results`
  - `Folder`
  - `Select all`
  - `Export`
  - `RIS format`
  - `Save`
- Speichern: Dateiname `gf20xx.txt`

IEEE Xplore

- Suchabfrage (Anführungszeichen nutzen für exakte Suche!):

  - Beispiel: `(("Author Affiliations": tech* univ* berlin) OR "Author Affiliations": "TU Berlin") OR "Author Affiliations": "Berlin Inst* of technol*")`
- Suchergebnisse filtern nach
  - Jahr
  - Dokumententyp: `document type = Journals & Magazines`
- Daten exportieren:
  - `Display all records on one page`
  - `Select All on Page`
  - `Download Citations`
  - `Output Format= RIS`
  - `Download`
- Speichern: Dateiname `ieee20xx.txt`

InSpec

- Suchabfrage: Namensvarianten für die eigene Institution (Anführungszeichen nutzen für exakte Suche!) und Jahresangabe
  - Beispiel: `"tech* uni* berlin" OR "TU Berlin" (Address) and 2014 (year)`
- Suchergebnisse filtern nach
  - Dokumententyp: `document type = "journal paper"`, dabei `"conference paper"` explizit ausschließen!
- Daten exportieren:
  - `Save to Other Formats`
  - `Number of Records = Records 1 to 500`
  - `Record Content = Full Record`
  - `File Format = Tab-delimited (Win, UTF-8)`
- Datenexport wiederholen für alle vorhandenen Datensätze (InSpec erlaubt Herunterladen von max. 500 Einträgen pro Durchgang)
- alle Ergebnisse in *eine* Datei kopieren (Spaltennamen der Einzeldateien nur einmal und als erste Zeile übernehmen!)
- Speichern: Dateiname `inspec20xx.txt`

Library and Information Science Abstracts (LISA)

- Suchabfrage (erweiterte Suche) via ProQuest:
  - Beispiel: `(ea(tu-berlin.de) OR af(Techn* berlin)) AND rtype.exact ("Article" OR "Journal Article") AND pd(2014)`
- Daten exportieren:
  - `Items per page: 100` (am Seitenende)
  - `Select 1-100` (wiederholen für alle Treffer)
  - `Save`
  - `Export/Save: RIS (works with EndNote, Citavi, etc.)`

    - Continue
    - Speichern
- Speichern: Dateiname lisa20xx.txt

ProQuest Social Sciences

- Suchabfrage (erweiterte Suche):
    - Beispiel: au(tech* univ* berlin) OR au(TU Berlin)
- Suchergebnisse filtern nach
    - Jahr
    - Dokumententyp: Source type = Scholarly Journals
- Daten exportieren:
    - Items per page: 100 (am Seitenende)
    - Select 1-100 (wiederholen für alle Treffer)
    - Save
    - Export/Save: RIS (works with EndNote, Citavi, etc.)
    - Continue
    - Speichern
- alle Ergebnisse in *eine* Datei kopieren (Spaltennamen der Einzeldateien nur einmal und als erste Zeile übernehmen!)
- Speichern: Dateinamen pq20xx.txt

PubMed

- Suchabfrage: Namensvariante für die eigene Institution (Anführungszeichen nutzen für exakte Suche!) und Jahr
    - Beispiel: (((((("Technische Universitat Berlin" [Affiliation]) OR "TU Berlin" [Affiliation]) OR "Tech Univ Berlin" [Affiliation]) OR "Berlin Institute of Technology" [Affiliation]) OR "Berlin Inst* Technol*" [Affiliation]) AND ("2014/01/01" [Date - Publication] : "2014/12/31" [Date - Publication])
- Daten exportieren:
    - Send to: File
    - Format: MEDLINE
- Speichern: Dateiname pubmed20xx.txt

SciFinder (CAplus)

- Suchabfrage:
    - Beispiel: "tech univ berlin" (company)
- Suchergebnisse filtern nach
    - Jahr: publication year = 20xx
    - Dokumententyp: document type = journal or review

  - Datenbank: `database = CAPLUS`
- Dubletten entfernen: `Tools: remove duplicates`
- Daten exportieren:
  - `Export`
  - `Range: 1-100`
  - `For: Citation Manager - Quoted Format (*.txt)`
  - `Details: Quote Character: "`
  - `Delimiter: tab-separiert`
- Datenexport wiederholen für alle vorhandenen Datensätze (SciFinder erlaubt Herunterladen von max. 100 Einträgen pro Durchgang)
- alle Ergebnisse in *eine* Datei kopieren (Spaltennamen der Einzeldateien nur einmal und als erste Zeile übernehmen!)
- Speichern: Dateiname `sf20xx.txt`

Scopus

- Suchabfrage:
  - Beispiel: `(AF-ID("Technische Universitat Berlin"60011604) OR (AFFIL(techn* AND univ* AND berlin)) OR (AFFIL(inst* AND technol* AND berlin)) AND DOCTYPE ( ar OR re ) AND PUBYEAR = 2014)`
- Daten exportieren:
  - `Select all`
  - `Export`
  - `RIS Format`
  - `Choose the information to export: All available information`
  - `Export`
- Datenexport wiederholen für alle vorhandenen Datensätze (Scopus erlaubt Herunterladen von max. 2000 Datensätzen pro Durchgang)
- alle Ergebnisse in *eine* Datei kopieren (Spaltennamen der Einzeldateien nur einmal und als erste Zeile übernehmen!)
- Speichern: Dateiname `scopus20xx.txt`

Sport Discus

- Suchabfrage (Search modes – Boolean/Phrase) via EBSCOhost:
  - Beispiel: `AF(Techn* univ* berlin OR TU Berlin)`
- Suchergebnisse filtern nach
  - Jahr: `20140101-20141231`
  - Dokumententyp: `Document Type: Article`
- Daten exportieren:
  - `Share`
  - `Add to Folder: Results`
  - `Folder`

  - `Select all`
  - `Export`
  - `RIS format`
  - `Save`
- Speichern: Dateiname `sd20xx.txt`

TEMA

- Suchabfrage: Namensvariante für die eigene Institution (Anführungszeichen nutzen für exakte Suche!) und Jahr
  - Beispiel: `"TU Berlin" (Institution), 2014 (Jahr)`
- Suchergebnisse filtern nach
  - Dokumententyp: `document type = Zeitschrift`
- Daten exportieren:
  - `Titel pro Seite`
  - `Alle auswählen`
  - `Auswahl Anzeigen`
  - `Alle auswählen`
  - `Auswahl als RIS speichern`
- Datenexport wiederholen für alle vorhandenen Datensätze (TEMA erlaubt Herunterladen von max. 100 Einträgen pro Durchgang)
- alle Ergebnisse in *eine* Datei kopieren (Spaltennamen der Einzeldateien nur einmal und als erste Zeile übernehmen!)
- Speichern: Dateiname `tema20xx.txt`

Web of Science Core Collection

- Suchabfrage: Namensvarianten für die eigene Institution und Jahresangabe
  - Beispiel: `technical university of berlin (organization enhanced) + 2014 (year)`
- Suchergebnisse filtern nach
  - Dokumententyp: `article or review (document type)`
- Daten exportieren:
  - `Save to Other Formats`
  - `Number of Records = Records 1 to 500`
  - `Record Content = Full Record`
  - `File Format = Tab-delimited (Win, UTF-8)`
- Datenexport wiederholen für alle vorhandenen Datensätze (WOS erlaubt Herunterladen von max. 500 Einträgen pro Durchgang)
- Ergebnisse in *eine* Datei kopieren (Spaltennamen der Einzeldateien nur einmal und als erste Zeile übernehmen!)
- Speichern: Dateiname `wos20xx.txt`

DOAJ

Das DOAJ stellt seine Daten zur Weiternutzung zur Verfügung (vgl. Hinweise unter https://doaj.org/faq#metadata), so kann u. a. eine CSV-Datei heruntergeladen werden: https://doaj.org/csv.

Damit die CSV-Datei vom Skript verarbeitet werden kann, muss sie unter dem Namen `doaj.txt` als tab-separierte Textdatei mit der Kodierung UTF-8 abgespeichert und in dem Unterordner `input-files` abgelegt werden. Die Umformatierung von comma-separiert in tab-separiert kann z. B. in MS Excel erfolgen; dabei ist zu beachten, dass die UTF-8-Kodierung nicht verändert werden darf. Beim Einlesen der CSV-Daten ist zudem darauf zu achten, dass die Spalten für ISSN und eISSN als `Text` eingelesen werden müssen, damit die Zeichenketten nicht irrtümlich als Datumsangaben interpretiert und in der Zeichenfolge verändert werden.

3.4 Datenbereinigung

Automatisierte Auswertungen sind nur so gut wie die zugrundeliegenden Daten. Um die Datenqualität zu erhöhen, empfiehlt sich eine nachträgliche Bearbeitung der Datei `allPubs.txt`. Die Bearbeitung kann in einem Texteditor oder Tabellenkalkulationsprogramm erfolgen. Empfohlen wird jedoch die Verwendung der Open-Source-Software OpenRefine[22].

Fehlerhafte DOIs Mitunter werden in den Datenbanken fehlerhafte DOIs gelistet. Es kann sich dabei um einfache Schreibfehler handeln (fehlende Zeichen, vertauschte Zeichen) oder auf Probleme bei der DOI-Registrierung zurückgehen. Die DOIs sollten manuell überprüft und ggf. korrigiert werden. Im Folgenden kann etwa Unpaywall erneut abgefragt werden, um weitere OA-Artikel zu identifizieren.

Identifikation weiterer OA-Artikel Mitunter findet Unpaywall mehrere OA-Versionen eines Artikels; alle OA-Versionen werden in dem Feld `oa_locations` angegeben. Die laut Unpaywall „beste“ OA-Version wird zusätzlich in dem separaten Feld `best_oa_location` angegeben. Das Python-Skript wertet lediglich die Angaben in diesem Feld `best_oa_location` aus und wertet in der Ausgabedatei solche Artikel als `hybrid`, für die die folgenden Parameter zutreffen:

- `oaDOI[is_oa] = True`
- `oaDOI[journal_is_oa] = False`
- `oaDOI[host_type] = publisher`
- `oaDOI[license] = 'cc'`

Ist ein Artikel zwar über die Verlagswebseite frei verfügbar, steht aber nicht unter einer Creative-Commons-Lizenz, wird er von dem Skript nicht als OA gewertet. Parallel dazu könnte auch eine

[22] OpenRefine ist für Windows, Mac und Linux verfügbar, Download unter http://openrefine.org/download.html. Es gibt zahlreiche Tutorials zur Verwendung online, vgl. etwa https://github.com/OpenRefine/OpenRefine/wiki/Recipes.

(„grüne“) OA-Version über ein Repositorium verfügbar sein. Dies wird in dem Skript momentan nicht berücksichtigt. Diese Artikel können in den Daten durch folgende Filter identifiziert werden, für die weitere Unpaywall-Abfragen durchgeführt werden könnten[23]:

- `OA-Status = None`
- `oaDOI[is_oa] = True`
- `oaDOI[journal_is_oa] = False`
- `oaDOI[host_type] = publisher`

Neben zusätzlichen „grünen“ Artikeln können auch weitere „hybride“ Artikel durch manuelle Prüfung ermittelt werden: Einige Verlage exponieren Lizenzangaben nur eingeschränkt oder gar nicht maschinenlesbar.[24] Bei der eigenen Datenanalyse zeichnete sich die Tendenz ab, dass sich für folgende Verlage eine manuelle Prüfung der Lizenz dann lohnt, wenn in den Unpaywall-Daten als Lizenzhinweis `implied OA` enthalten ist: Royal Society of Chemistry[25], Oxford University Press, Cambridge University Press, Institute of Mathematical Statistics. Für die manuelle Verifizierung kann entweder die Datei `allPubs.txt` (Feld: `oaDOI[license]`) oder die Datei `oaDOI-response.txt` (Feld: `license`) herangezogen werden.

Verifizierung der identifizierten Institutionen Enthalten die Datenbankdaten in den Affiliationsangaben mehrere Einträge (z. B. bei geteilter Korrespondenzautorschaft), kann es leicht zu *false positives* kommen. Enthält das Affiliationsfeld bspw. die Angaben `FU Berlin` und `Tech Univ Dresden`, werden vom Skript mit den momentanen Namensvarianten fälschlicherweise die FU Berlin und TU Berlin als Institutionen der KorrespondenzautorInnen erkannt. Es empfiehlt sich daher, die Artikel manuell zu überprüfen für die in der Spalte `found name variant` mehrere Einträge (durch Semikolon getrennt) aufgeführt werden. Sollen nicht nur Affiliationsangaben für die Korrespondenzautor*innen, sondern für alle Autorinnen und Autoren ausgewertet werden, sollten auch die Mehrfacheinträge in der Spalte `all identified name variants` überprüft werden.

Identifikation von weiteren Artikeln mit Korrespondenzautorschaft Die Korrespondenzautorschaft wird auf Basis der im dritten Abschnitt des Skripts angelegten Namensvarianten identifiziert. Die Spalte `corresponding author` (bzw. `affiliations` für die vorhandenen Affiliationsangaben aller Autorinnen und Autoren) könnte nach weiteren Namensvarian-

[23] Mithilfe von OpenRefine kann dies sehr einfach erfolgen, vgl. die Anleitung *Collecting Open Access information using OpenRefine and the oaDOI API (special bonus: the DOAJ API)* (http://www.libraryworkflowexchange.org/wp-content/uploads/2017/07/Collecting-Open-Access-information-using-OpenRefine-and-the-oaDOI-API.pdf).

[24] D. h. einige Verlage geben weder auf der eigenen Webseite die Lizenz in einem maschinenlesbaren Format (z. B. als Meta-Element im HTML-Header oder als RDFa-Element bei Angabe der Lizenz auf der Seite) noch in den Crossref-Metadaten an. Die freie Lizenz kann in solchen Fällen lediglich über HTML-Scraping automatisch identifiziert werden.

[25] RSC exponiert sehr wohl die Lizenz-URL in den Crossref-Metadaten, allerdings in dem nicht-kanonischen Feld `assertion` (statt in dem dafür vorgesehen Feld `license`. Die Überprüfung könnte also automatisiert werden, indem die Crossref-Schnittstelle abgefragt und der Inhalt des Feldes `assertion` analysiert wird.

ten (inkl. üblicher Schreibfehler) durchsucht werden. Es empfiehlt sich zudem die Suche nach der E-Mail-Domain der jeweiligen Einrichtung in der Spalte `email`.

Dublettencheck Der Dublettencheck erfolgt in dem Skript auf Basis von DOIs bzw. Angaben zu Autor/Titel (vgl. Hinweise unter 2.1 auf S. 5 bzw. 2.6 auf S. 11). Eine vollständige Dublettenerkennung ist unwahrscheinlich. Mithilfe von OpenRefine könnten potentielle Dubletten mithilfe einer Facette[26] aufgerufen, manuell überprüft und ggf. gelöscht werden.

Vereinheitlichung Verlage und Journale In den Datenbanken sind mitunter verschiedene Varianten für Journale und Verlage hinterlegt. Will man eine statistische Auswertung nach Verlagen und/oder Journalen vornehmen (z. B. Verteilung auf Verlage, Anzahl Journale insgesamt, usw.), sollten diese Angaben vereinheitlicht werden.

[26] Zur Umsetzung vgl. den Eintrag *How do I find duplicates in a column?* unter https://github.com/OpenRefine/OpenRefine/wiki/FAQ.

Anhang

Tabelle 1: Übersicht der einzelnen Skriptdateien

Skriptdatei	Erläuterung
main.py	Hauptskript
RIS-fields.csv	Übersicht RIS-Felder der verschiedenen Datenbanken

Tabelle 2: Übersicht der vom Skript einzulesenden Dateien (Unterordner `input-files`)

Eingabedatei	Erläuterung
docsChecked.txt	Publikationen, für die Affiliation der Erst- bzw. Korrespondenzautorschaft manuell geprüft wurde
doaj.txt	DOAJ-Metadaten
bsc20xx.txt	Artikeldaten Business Source Complete
cab20xx.txt	Artikeldaten CAB Abstracts
cinahl20xx.txt	Artikeldaten CINHAL
ebsco20xx.txt	Artikeldaten Academic Search Premier (EBSCO)
embase20xx.txt	Artikeldaten Embase
gf20xx.txt	Artikeldaten GeoRef
ieee20xx.txt	Artikeldaten IEEE Xplore
inspec20xx.txt	Artikeldaten InSpec
lisa20xx.txt	Artikeldaten LISA
pq20xx.txt	Artikeldaten ProQuest Social Sciences
pubmed20xx.txt	Artikeldaten PubMed
scopus20xx.txt	Artikeldaten Scopus
sd20xx.txt	Artikeldaten Sport Discus
sf20xx.txt	Artikeldaten SciFinder
tema20xx.txt	Artikeldaten TEMA
wos20xx.txt	Artikeldaten Web of Science

Tabelle 3: Übersicht der wichtigsten im Skript verwendeten Variablen

Variable im Skript	Erläuterung
checkToDo	Skriptfunktionalität (de-)aktivieren: manuelle Prüfung Erst-/Korrespondenzautorschaft
contactCR	Skriptfunktionalität (de-)aktivieren: Abfragen der API von Crossref
contactOaDOI	Skriptfunktionalität (de-)aktivieren: Abfragen der API von Unpaywall
doAnalysis	Skriptfunktionalität (de-)aktivieren: Datenauswertung (Statistik, Diagramme)
doReadIn	Skriptfunktionalität (de-)aktivieren: Einlesen Artikeldaten aus Datenbankrecherchen
finalList	Liste aller gefundenen Artikel
yearMin	Start des Untersuchungszeitraums; es ist das Jahr mit vier Stellen anzugeben (relevant für korrekte Erkennung der PubMed-Daten)
yearMax	Ende des Untersuchungszeitraums, es ist das Jahr mit vier Stellen anzugeben (relevant für korrekte Erkennung der PubMed-Daten)

Tabelle 4: Übersicht der vom Skript ausgegebenen Dateien (Unterordner `output-files`)

Ausgabedatei	Erläuterung
allPubs.txt	Liste aller gefundenen Artikel
docsCheckedCantFind.txt	Hilfsdatei, wird nur dann erstellt, wenn beim Einlesen der Datei `docsChecked.txt` ein Fehler auftritt. (z. B. aufgrund eines Copy/Paste-Fehlers bei der Erstellung der txt-Datei)
docsToBeChecked.txt	Liste der Publikationen, für die Affiliation der Erst- bzw. Korrespondenzautorschaft manuell zu prüfen ist
finalList	interne Arbeitsdatei aller Artikeldaten
oaDOI-response.txt	Werte aus Unpaywall-Schnittstelle
statistics_OA.txt	Statistik der analysierten Artikel
statistics_goldPublishers.txt	Häufigkeitsverteilung der Gold-Artikel (OA-Artikel in OA-Zeitschriften) auf Verlage
DOIs-oaDOI-error.txt	Liste der DOIs, für die Unpaywall eine Fehlermeldung zurückgegeben hat

Tabelle 5: Angaben in allPubs.txt

Feld	Quelle(n)	Erläuterung
authors	Datenbanken	Angaben gekürzt auf max. 2500 Zeichen
title	Datenbanken	
OA-Status	DOAJ, Unpaywall	Angaben, über welchen Weg Artikel OA verfügbar ist (`None`, `gold`, `hybrid` oder `green`)
DOI	Datenbanken	
journal	Datenbanken	
ISSN	Datenbanken, Crossref	ISSN der Print- oder Onlineausgabe (wahrscheinlich Print)
eISSN	Datenbanken, Crossref	ISSN der Print- oder Onlineausgabe (wahrscheinlich Online)
publisher	Datenbanken, DOAJ, Unpaywall	für OA-Artikel werden ursprüngliche Angaben zwecks Vereinheitlichung überschrieben
year	Datenbanken	PubMed indiziert verschiedene Daten: PubMed-Suche deckt alle Datumsfelder ab, während das Python-Skript nur ein Datumsfeld auswertet (`DP = Date of Publication`)
affiliations	Datenbanken	Affiliationen der Autorinnen und Autoren
all identified name variants	Skript	Institutionskürzel für alle vorhandenen Affiliationen, auf Basis der im Skript angesetzten Institutionsnamen zugeordnet wurden
corresponding author	Datenbanken	Affilation der Korrespondenzautorin bzw. des Korrespondenzautors
found name variant	Skript, ggf. manuell	Institutionskürzel für Affiliation der Korrespondenzautorschaft, das auf Basis der im Skript angesetzten Institutionsnamen zugeordnet wurde
e-mail	Datenbanken	
subject	Datenbanken	Angaben aus Web of Science, SciFinder
DOAJ subject	DOAJ	im DOAJ werden Zeitschriften nach der Library of Congress Classification klassifiziert
funding	Datenbanken	Angaben aus Web of Science
license	DOAJ, Unpaywall	Standardlizenz für Journal laut DOAJ bzw. Lizenzangabe für Artikel aus Unpaywall
databaseID	Skript	
notes	Skript	verschiedene Indikatoren für Datenquellen: `Checked by hand.` = Affiliation für Korrespondenzautorschaft manuell ermittelt; `Identified via DOAJ` = OA-Status ermittelt mithilfe von DOAJ; `Identified via Unpaywall` = OA-Status ermittelt mithilfe von Unpaywall
oaDOI[is_oa]	Unpaywall	`True` = Artikel ist OA verfügbar
oaDOI[journal_is_oa]	Unpaywall	`True` = Journal ist OA-Zeitschrift
oaDOI[host_type]	Unpaywall	primäre OA-Version verfügbar über Verlag (`publisher`) oder gesichertes Repositorium (`repository`)
oaDOI[license]	Unpaywall	
APC Amount	DOAJ	
APC Currency	DOAJ	