Benutzerhandbuch Formalerschliessungungs-Applikation FID II

Angewandte Computerlinguistik - Goethe Universtät Frankfurt

Frank Abromeit

7. Februar 2022

Inhaltsverzeichnis

1 Extraktion von Metadadaten

Die Applikation ermöglicht die Extraktion der formalen Metadaten wie Titel, Author, etc. aus PDF-Dokumenten. Im Prinzip werden beliebige Dokumentenlayouts unterstützt. Für die Extraktion ist es notwendig eine Beschreibung der zu extrahierenden Daten anzugeben, die in einer Konfigurationsdatei eingetragen werden müssen. Die Konfiguration ist jeweils unterschiedlich für verschiedene Layouts. Der Aufwand für standardisierte Layouts ist i.d.R. gering. Für einige Layouts kann der Aufwand für die Erstellung der Beschreibung der Extraktionsparameter aber hoch und ineffizient sein. In der Praxis hat sich gezeigt, dass es besser ist sich auf einige standardisierte Layouts zu konzentrieren, da auch der Aufwand für Nachbearbeitungen sonst zu hoch ist.

1.1 Beschreibung der Extraktionsverfahren

Das Ziel der Extraktion ist das Auffinden von Metadaten in einer Publikation wie Titel, Author, Jahr, Zeitschrift, Ausgabe. Dazu werden die PDF-Dokumente, die einzelne Aufsätze oder auch Aufsatzsammlungen enthalten können, zunächst in das XML-Format konvertiert. Um die Extraktionsverfahren besser zu verstehen ist es nützlich zu wissen wie der PDF-Text in XML repräsentiert wird. Dazu folgendes Beispiel (siehe Listing 1), das auch gleichzeitig einen möglichen Problemfall für die Metadatenextraktion zeigt. Die <fontspec> Zeilen zu Beginn definieren die verwendeten Schriftarten. In den darauffolgenden <text> Zeilen ist der Text jeweils mit den Positionsparametern **top** und **left** sowie den Formalparametern für Schriftart (**font**) und Schriftgrösse (**heigth**) sowie optionalen oder Markierern für fette oder kursive Schrift dargestellt. Auch deswegen kann es sein, dass ein Wort im Text in mehrere XML <text> Zeilen aufgeteilt ist. Im Beispiel unten trifft das auf den Autorennamen (Carl Pollay) zu, da dieser für die Anfangsinitialen jeweils eine andere Schriftgrösse und Schriftart verwendet. Dieser Fall ist problematisch für die Erkennung und führt zu Fehlern bei der Erkennung.

Listing 1: XML Darstellung für PDF-Dokument

```
<page number="3" position="absolute" top="0" left="0" height="1188" width="918">
<fontspec id="5" size="18" family="Times" color="#000000"/>
<fontspec id="6" size="11" family="Times" color="#000000"/>
<fontspec id="7" size="16" family="Times" color="#000000"/>
<fontspec id="8" size="9" family="Times" color="#000000"/>
<fontspec id="9" size="11" family="Times" color="#000000"/>
<text top="109" left="229" width="470" height="23" font="5">
THE EMERGENCE OF MANDARIN METAPHORS </text>
<text top="133" left="362" width="200" height="23" font="5">
FOR THE INTERNET </text>
<text top="157" left="459" width="410" height="20" font="2"></text>
<text top="178" left="412" width="11" height="18" font="0">C</text>
<text top="181" left="423" width="30" height="15" font="6">ARL </text>
<text top="178" left="453" width="9" height="18" font="0">P</text>
<text top="181" left="462" width="44" height="15" font="6">OLLEY</text>
<text top="178" left="507" width="4" height="18" font="0"> </text>
```

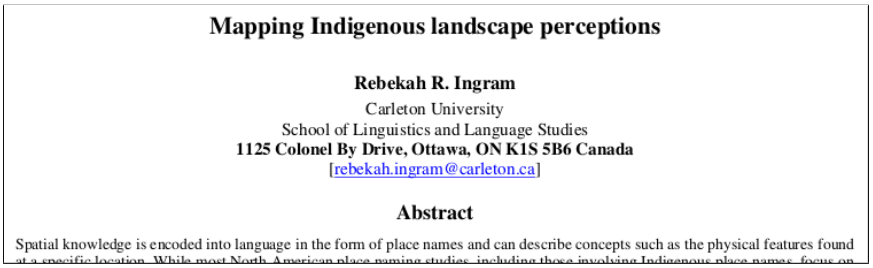
```
<text top="196" left="459" width="5" height="20" font="7"><b> </b></text>
<text top="217" left="162" width="5" height="20" font="7"><b> </b></text>
<text top="238" left="162" width="598" height="16" font="3">
Conceptual metaphors underlie many of the everyday expressions we use
when describing novel
</text>
```

1.2 Extraktion von Titel und Author(en) Information in Einzelaufsätzen

Es sind zwei Extraktionsverfahren implementiert, ein automatisches, bei der nur wenige Parameter gesetzt werden müssen, und ein manuelles, das es erfordert, dass man die XML-Struktur für eine PDF-Dokumentklasse anschauen muss, um die korrekten Parameter für die verwendete Schriftgrösse und Schriftart für Titel, Author, etc. in der Extraktionskonfiguration eintragen zu können. Das ist zum einen sehr aufwendig, da eventuell bis zu 10 Parameter gesetzt werden müssen, die meistens auch erst nach vielen Testläufen genau bestimmt sind, aber auch nicht zuverlässig, wenn z.B. die Schriftgrösse und Schriftart für Titel und Author innerhalb der Dokumente einer Datenquelle variieren. Das manuelle Verfahren wird wegen dieser Nachteile hier nicht weiter beschrieben.

Das automatische Verfahren macht folgenden Annahmen über ein zu parsendes PDF-Dokument:

- Die Seitennummer auf der sich Titel und Author befinden ist über die verschiedenen Dokumente einer Datenquelle immer gleich. Diese wird mit dem Parameter **firstPage** in der Konfiguration angegeben.
- Die Seite beginnt entweder mit dem Titel oder einer Headerzeile, in der z.B. weitere Information wie der Zeitschriftentitel, Jahr, etc. oder auch Seitenzahlen stehen. Wenn eine Headerzeile vorhanden ist, dann muss diese in der Konfiguration angegeben werden (siehe Listing 3), da diese sonst fälschlicherweise als Titel angenommen wird.
- Nach dem Titel folgt die Autoreninformation, eventuell zusammen mit Zusatzinformationen wie Email-Adresse und Ortsangaben


Mapping Indigenous landscape perceptions

Rebekah R. Ingram
Carleton University
School of Linguistics and Language Studies
1125 Colonel By Drive, Ottawa, ON K1S 5B6 Canada
[rebekah.ingram@carleton.ca]

Abstract

Spatial knowledge is encoded into language in the form of place names and can describe concepts such as the physical features found

Abbildung 1: Titel und Author in Einzelaufsatz

Die Extraktionsbeschreibung in Listing 2 kann für Dokumente verwendet werden, deren Layout dem in Abb. 1) entspricht. Für den Titel und den Author werden dort verschiedene Schriftgrössen verwendet. Da Titel und Author(en) jeweils mehrere Zeilen umfassen können gruppiert ein Algorithmus gleichformatierte aufeinanderfolgende Zeilen und kann diese gemäß der vordefinierten Reihenfolge (oben) dem Titel, dem/den Author(en) bzw. Kopf-/Fusszeile zuordnen. Mit dem Parameter **bindSequentialLinesBy** kann eine Strategie zur Gruppierung aufeinanderfolgender Zeilen festgelegt werden. Zulässige Werte sind **fontSize**, **fontType**, sowie **fontSize+fontType**. Der Parameter *fontType* schliesst dann alle Formalparameter (Schriftart, Fettschrift, Kursivschrift) mit ein.

Listing 2: Titel und Author Extraktion

```
{
"id" : "124671918",
"subId" : "a1",
"extractorConfig" : {
"bindSequentialLinesBy" : "fontSize",
"firstPage" : 2}}
```

Für den Fall, dass auch Kopf-/Fusszeilen Informationen vorhanden sind (siehe Abb. 2) enthält die Extraktionsbeschreibung im Feld **footerDescription** die Extraktionsparameter für die Angaben dort, wie z.B. den Namen einer Zeitschrift sowie mögliche weitere Informationen wie Jahrgang, Jahr, Ausgabe, Seitenzahlen, etc. . Ein Beispiel dazu findet sich in Listing 3.

1.3 Reguläre Ausdrücke

Die Beschreibung der Position der zu extrahierenden Metadaten in einer Kopf-/Fusszeile wird über reguläre Ausdrücke vorgenommen. Reguläre Ausdrücke beschreiben in der Informatik formale Sprachen, und lassen sich dazu verwenden um Muster in einer Zeichenkette zu erkennen, und diese daraus zu extrahieren. Es ist erwähnenswert, das die Sprache der regulären Ausdrücke äquivalent zur den von Endlichen Automaten erkennbaren Sprachen ist. Ein wesentliches Merkmal regulärer Ausdrücke sind Quantoren, die die Anzahl für das Vorkommen einer bestimmten Zeichenkette in einer Texteingabe beschreiben. Dazu gehören * (Kleene-Operator) *kein-Mal oder beliebig oft*, + *mindestens einmal*, ? *keinmal oder einmal*. Eine andere Möglichkeit für die Festlegung einer Anzahl bietet der geklammterte Ausdruck {min,max}. Daneben gibt es vordefinierte Zeichenklassen (siehe auch `https://de.wikipedia.org/wiki/Regulärer_Ausdruck`), z.B.:

- \d Zahlen
- \s Leerzeichen
- . Beliebige Zeichen

Will man nach Zeichen suchen, die auch Bestandteil der Beschreibungssprache der regulären Ausdrücke sind, so müssen diese (escaped) werden, so wird z.B. aus '.' => '\.' .

Beispiel für einen regulären Ausdruck, der die Zeichenkette '1.10.2008' erkennen kann: "1\.10 \.2008". Wenn man generell nach Daten der obigen Form sucht dann geht das mit dem Ausdruck "\d+\.\d\.\d". Diese Beschreibung ist aber möglicherweise zu allgemein, da damit auch die Zeichenkette '123.234.235' erkannt wird. Ein anderes Problem entsteht wenn zustätzliche Leerzeichen vorhanden sind, wie in '1. 10. 2008'. Der folgende Ausdruck schliesst diese Fälle mit ein: "\d{1,2}\s*\.\s*\d{1,2}\s*\d{4}".

1.4 Information in Kopf- und Fusszeilen

Eine Footerkonfiguration enthält eine Beschreibung der in einer Kopf/Fusszeile enthaltenen Informationen. Das Extraktionsverfahren beruht auf dem Verfahren, der bereits im Kapitel 1.3 beschriebenen regulären Ausdrücke. Da reguläre Ausdrücke ab einer gewissen Parameteranzahl schwer zu lesen, und auch zu verstehen sind, wurden einige Vereinfachungen vorgenommen, die die entstehenden Ausdrücke kürzer und auch für Laien besser verständlich machen sollen. Das bedeutet, dass der Benutzer mit vordefinierten Schlüsselworten arbeitet, die von der Applikation in reguläre Ausdrücke übersetzt werden. Die vordefinierten Schlüsselworte sind

- **title** Titel eines Artikels
- **author** Author(en)
- **year** Erscheinungsjahr
- **journalTitle** Name einer Publikation
- **volume** Band
- **issue** Ausgabe
- **beginPage** Anfangsseite eines Artikels
- **endPage** Letzte Seite eines Artikels
- **date** Datumsangabe
- **place** Ortsangabe

Zudem können in der Footerkonfiguration auch direkt ein oder mehrere reguläre Ausdrücke angegeben werden. Schliesslich wird in der Beschreibung, ebenfalls mit einem regulären Ausdruck festgelegt, wie die Fusszeile gefunden werden soll. Das geschieht dadurch das man z.B. einen markanten Präfix des Texts in der Fusszeile, der immer gleich ist, mit dem Parameter **footerStartsWithText** festlegt. Wenn verschieden von **firstPage** kann mit **footerPage** eine Seite angegeben werden auf der sich der Footer befindet. Das Finden des richtigen regulären Ausdrucks ist für Laien nicht trivial und erfordert etwas Übung. D.h. man muss ausprobieren was am Besten funktioniert. Dazu lässt man die Applikation wiederholt laufen und kontrolliert die Ausgabe.

Listing 3: Parameter für Fusszeile

```
{
"id" : "048593834",
"extractorConfig" : {
"bindSequentialLinesBy" : "fontType",
"firstPage" : 1,
"footerDescriptions" : [
{"footerStartsWithText" : "McGill Working Papers in Linguistics"},
"footerDescription" : "journalTitle , Volume @volume@ , Issue @issue , beginPage-endPage , year"},
{"footerStartsWithText" : "McGill Working Papers in Linguistics",
"footerDescription" : "journalTitle , Volume @volume@ . issue , Winter @year"}
]}
}
```

Erläuterung der Fusszeilen-Beschreibung aus Listing 3 für das PDF in Abb. 4:

footerDescription : "**journalTitle**, Volume @**volume**@.**issue**,Winter @**year**"

McGill Working Papers in Linguistics, Volume 22.1 Winter 2012

Die Beschreibung enthält die Schlüsselworte **journalTitle**, **volume**, **issue** und **year**. Andere Symbole wie ',' und Zeichenketten wie 'Winter' dienen im Prinzip nur dazu um zu definieren wo eine Texteinheit anfängt bzw. endet. Zum Beispiel wird definiert, dass nach dem Zeitschriftentitel ein Komma stehen muss. Das bedeutet dann, dass alles was vor dem ersten Komma in einer Zeile, die mit 'McGill Working Papers in Linguistics' beginnt als Zeitschriftentitel (**journalTitle**) angenommen wird. Nach dem Komma steht 'Volume' in der Beschreibung, was wörtlich den vorhandenen Text in der Fusszeile sucht. Das darauffolgende @ ist eine Abkürzung für den regulären Ausdruck \s+' und dient dazu die **volume** Information herauszuparsen. Als nächstes wird definiert, dass vor der Information der Ausgabe (**issue**) ein Punkt und danach ein Komma stehen muss, gefolgt von 'Winter', worauf mit mind. einem Leerzeichen Abstand eine Jahresangabe (**year**) steht. Zusammengefasst : Schlüsselworte dienen dazu Textbestandteilen anhand ihrer Position eine Bedeutung zuzuordnen. Markante Zeichenketten aus der Fusszeile wie z.B. Satzzeichen (,.:!= oder auch Zeichenketten werden dazu verwendet um die Grenzen einer Texteinheit zu defnieren.

Für die Notation der Footerbeschreibung gelten folgende Regeln:

- Leerzeichen in der Beschreibung spielen keine Rolle. Im Beispiel oben ist es egal, ob 'journalTitle,' oder 'journalTitle ,' notiert ist.
- Hinter jedes Schlüsselwort wird intern automatisch der reguläre Ausdruck \s* angehängt.

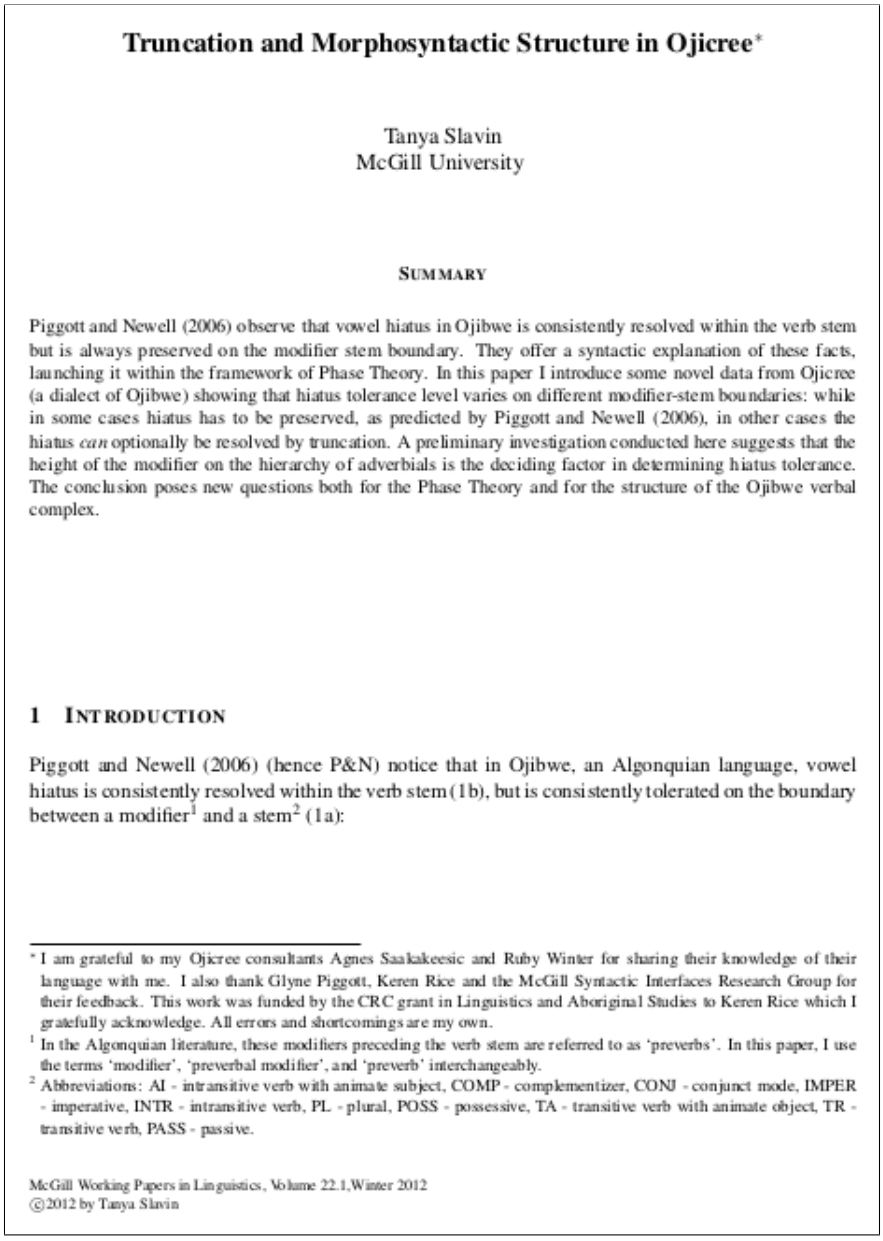
- Hinter jedes nicht-Schlüsselwort, im Beispiel : ‚@.Volume Winter wird der reguläre Ausdruck \s* angehängt.
- An das Ende des regulären Ausdrucks wird \.* angehängt. Damit wird erreicht, dass eine Beschreibung erkannt wird, auch wenn noch weiterer Text folgt. Der angegeben Ausdruck würde deshalb auch für den Text 'McGill Working Papers in Linguistics, Volume 22.1 Winter 2012, Frankfurt/Main' funktionieren.
- Es können in der Beschreibung auch reguläre Ausdrücke direkt angegeben werden. Diese müssen in einfachen Anführungszeichen eingeschlossen sein. So wäre eine alternative Schreibweise desselben Ausdrucks: **journalTitle** '\,\s*Volume\s+' **volume**'\s+\.\s*' **issue** '\s*\,\s*Winter\s+' **year**
- Alle Symbole in einer Beschreibung, die zur Metasprache der regulären Ausdrücke selbst gehören wie ‚.-?!()[] etc. werden automatisch 'escaped'.
- Es können mehrere Beschreibungen für eine Fusszeile angegeben werden, da Abweichungen in der Notation in einem PDF, z.B. wenn ein Jahrgang einer Zeitschrift eine andere Fusszeilenformatierung verwendet, dazu führt, dass die Erkennung fehlschlägt. Alternativ dazu kann man versuchen eine möglichst universelle Beschreibung in einem Ausdruck zusammenzufassen.
- Durch anhängen von ? an ein Schlüsselwort, z.B. journalTitle? wird erreicht, dass die Auswertung des regulären Ausdrucks von journalTitle ('.*?') nicht mehr *greedy* erfolgt. Das gilt nicht für Schlüsselworte die Zahlen einlesen (**beginPage**, **endPage**, **year**, **issue** und **volume**), da diese immer *greedy* sein müssen.

1.5 Zitierangaben im Text

Es kommt vor, das Aufsätze direkt eine Angabe enthalten, wie eine Publikation zitiert werden soll. In dieser Angabe stehen dann alle Metadaten direkt zusammen, was die Extraktion erleichtert. Für diesen Fall wurde eine eigenes Extraktionsverfahren definiert, welches sich stark an dem für Fusszeilen orientiert. D.h. man legt zuerst fest wo eine Zitierangabe beginnt, und aufhört. Genau wie bei einer Fusszeile gibt man in einer Beschreibung an wie diese formatiert ist.

Listing 4: Parameter für Zitierangaben

```
"id" : "124671918",
"subId" : "a3",
"extractorConfig" : {
"citationDescriptions" : [
{"citationStartsAfterText" : "Cite␣this␣item:",
"citationEndsBeforeText" : "Link␣to␣this␣item:",
"citationVariants" : [
{"contentDescription" : "author␣(year)␣title␣'\\..*'␣In␣journalTitle
␣␣edited␣'.*?'␣beginPage␣–␣endPage"}
]}],
```



Truncation and Morphosyntactic Structure in Ojicree*

Tanya Slavin
McGill University

SUMMARY

Piggott and Newell (2006) observe that vowel hiatus in Ojibwe is consistently resolved within the verb stem but is always preserved on the modifier stem boundary. They offer a syntactic explanation of these facts, launching it within the framework of Phase Theory. In this paper I introduce some novel data from Ojicree (a dialect of Ojibwe) showing that hiatus tolerance level varies on different modifier-stem boundaries: while in some cases hiatus has to be preserved, as predicted by Piggott and Newell (2006), in other cases the hiatus *can* optionally be resolved by truncation. A preliminary investigation conducted here suggests that the height of the modifier on the hierarchy of adverbials is the deciding factor in determining hiatus tolerance. The conclusion poses new questions both for the Phase Theory and for the structure of the Ojibwe verbal complex.

1 INTRODUCTION

Piggott and Newell (2006) (hence P&N) notice that in Ojibwe, an Algonquian language, vowel hiatus is consistently resolved within the verb stem (1b), but is consistently tolerated on the boundary between a modifier[1] and a stem[2] (1a):

* I am grateful to my Ojicree consultants Agnes Saakakeesic and Ruby Winter for sharing their knowledge of their language with me. I also thank Glyne Piggott, Keren Rice and the McGill Syntactic Interfaces Research Group for their feedback. This work was funded by the CRC grant in Linguistics and Aboriginal Studies to Keren Rice which I gratefully acknowledge. All errors and shortcomings are my own.

[1] In the Algonquian literature, these modifiers preceding the verb stem are referred to as 'preverbs'. In this paper, I use the terms 'modifier', 'preverbal modifier', and 'preverb' interchangeably.

[2] Abbreviations: AI - intransitive verb with animate subject, COMP - complementizer, CONJ - conjunct mode, IMPER - imperative, INTR - intransitive verb, PL - plural, POSS - possessive, TA - transitive verb with animate object, TR - transitive verb, PASS - passive.

McGill Working Papers in Linguistics, Volume 22.1,Winter 2012

Abbildung 2: Dokument mit Fusszeile

```
"firstPage" : 1
}
```

Mit den Attributen **citationStartsAfterText** und **citationEndsBeforeText** sowie mit dem Attribut **firstPage** wird die Position einer Zitierangabe festgelegt. Mit **citationVariants** gibt es hier auch die Möglichkeit mehrere Beschreibungen für einen Zitierstil festzulegen, z.B. für den Fall, dass sich dieser über verschiedene Ausgaben hin-

weg ändert. Die Definition des Zitierstils wird mit dem Parameter **contentDescription**, genauso wie bereits für Fusszeilen beschrieben, festgelegt.

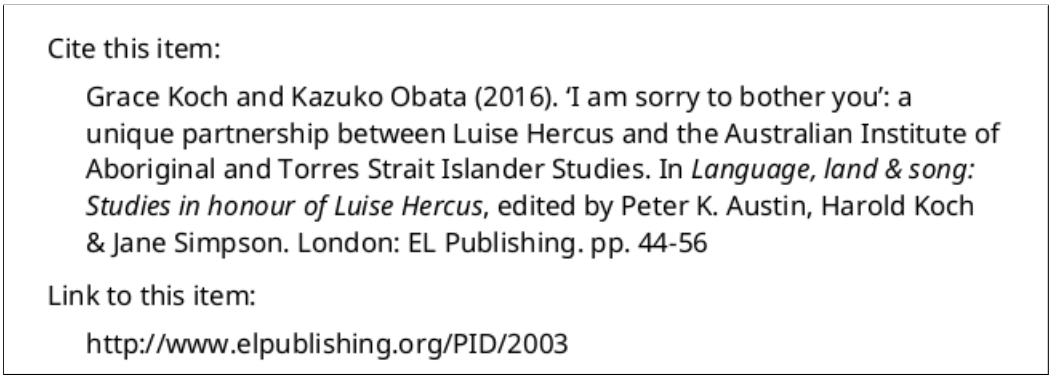


Abbildung 3: Dokument mit Zitierinformation

1.6 Inhaltsverzeichnisse in Aufsatzsammlungen

Aufsatzsammlungen enthalten in der Regel ein Inhaltsverzeichnis, was sich sehr gut für die Extraktion der Metadataen eignet. Aufsatzsammlungen ohne Inhaltsverzeichnis werden leider nicht unterstützt. Das Extraktionsverfahren ist im Prinzip eine Kombination der Verfahren mit regulären Ausdrücken und dem vollautomatischen Verfahren für Einzelaufsätze (siehe Kap. 1.2).

Listing 5: Extraktion aus einem Inhaltsverzeichnis

```
{
"id" : "mult_048020990",
"extractorConfig" : {
"tocDescriptions" : [
{"tocStartsAfterText" : "Table␣of␣Contents", "tocEndsBeforeText" : "Proceedings␣of␣the",
"tocEntryDescription" : [
{"contentDescription" : "title␣'[\\.,\\_]+'␣beginPage"},{"contentDescription" : "author"}
]}],
"bindSequentialLinesBy" : "fontType",
"firstPage" : 5,
"footerPage": 7,
"footerDescriptions" : [
footerStartsWithText"␣:␣"Proceedings of the 24th NWLC"},
{"footerDescription"␣:␣"journalTitle? ',.*?' year ',' place",␣"}]
}
```

Ein Inhaltsverzeichnis besteht aus jeweils gleichformatierten Textblöcken, die in der Regel zumindest die Titel und Autor(en) Information für einen Aufsatz enthalten. Mit der Angabe der Seite (**firstPage**) und der Position (**tocStartsAfterText** und **tocEndsBeforeText**) werden Anfang und Ende definiert. Im Parameter **tocEntryDescription** wird der Aufbau eines Eintrags im Inhaltsverzeichnis mit einer geordneten Liste [1]von **contentDescription** Feldern definiert. Jeder Eintrag dieser Liste bezieht sich auf

[1]d.h. die Reihenfolge spielt eine Rolle

ein oder mehrere Metadatenfelder, die jeweils die gleiche Formatierung (Schriftgrösse, Schriftart, Fett, Kursiv) haben. Welche Formatierung für die Gruppierung verwendet werden soll wird durch den Wert des Parameters **bindSequentialLinesBy** bestimmt. Für jede **contentDescription** wird dann mit einer Beschreibung in der bereits beschriebenen Syntax die Extraktion der Metadaten gesteuert. Auch hier besteht die Möglichkeit mehr als eine Beschreibung für ein Inhaltsverzeichnis anzulegen. Das Verfahren kann auch mit einer Fusszeilenerkennung kombiniert werden (siehe Listing 5). In diesem Fall werden die extrahierten Metadaten aus beiden Verfahren zusammengeführt.

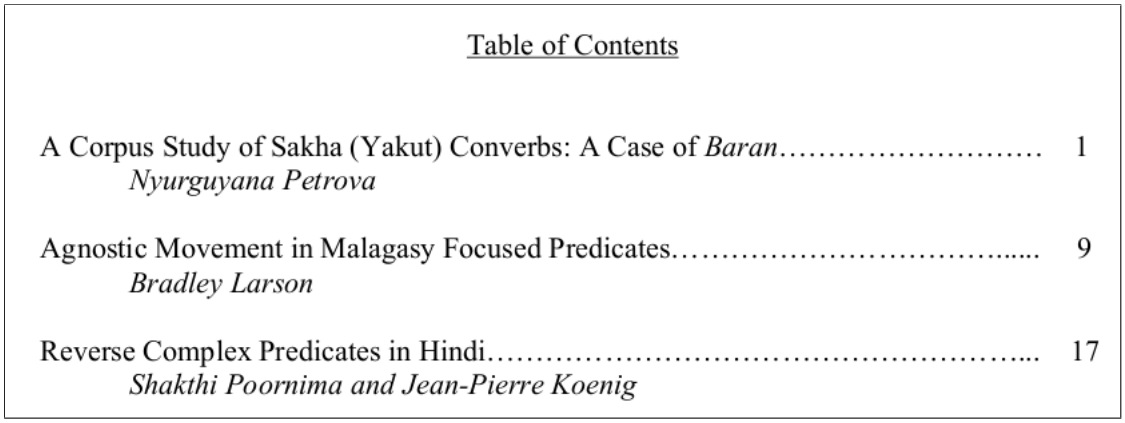


Table of Contents

A Corpus Study of Sakha (Yakut) Converbs: A Case of *Baran*..... 1
Nyurguyana Petrova

Agnostic Movement in Malagasy Focused Predicates..... 9
Bradley Larson

Reverse Complex Predicates in Hindi..... 17
Shakthi Poornima and Jean-Pierre Koenig

Abbildung 4: Dokument mit Inhaltsverzeichnis

1.7 Erkennung von Vor/Nachnamen

Die Findung des Vornamen bzw. Nachnamens für einen gegebenen Namensstring ist nicht trivial. Zwar existieren Namensammlungen für einzelne Sprachen. Da diese Datensätze i.d.R. nicht vollständig sind, können diese nur bedingt für die Trennung in Vor- und Nachnamen verwendet werden. Das aktuelle Verfahren verwendet Namensdefinitionen von bekannten Autorennamen der UB. Es können jedoch auch andere Namensdaten verwendet werden, die als CSV oder TSV Datei vorliegen müssen. Die Erkennung funktioniert so, dass ein Namensteil, der in der Liste der bekannten Nachnamen gefunden wurde, ausserdem nicht in der Liste der bekannten Vornamen vorhanden sein darf, damit ein Name als Nachname zugewiesen wird. Da Nachnamen auch Vornamen sein können sind Fehler bei Namensausrücken, die aus mehr als 2 Teilen bestehen unvermeidbar. Als zweite Möglichkeit für die Erkennung von Nachnamen werden bekannte Präfixe von Nachnamen verwendet (z.B. *von*, *Mc*, etc.). Diese sind als kommagetrennte Liste im Konfigurationsparameter *familyNamePrefixes* abgespeichert und können ebenfalls erweitert werden.

1.8 Erkennung von Untertiteln

Ein Untertitel muss mit einem Trennsymbol markiert sein. Dabei werden Trennsymbole, die gelöscht (z.B. Bindestrich) oder beihehalten (z.B. Fragezeichen) werden sollen, unterschieden. Diese sind in den Konfigurationsparametern *subTitleSplitRegexRemove* und *subTitleSplitRegexMaintain* als reguläre Ausdrücke definiert.

1.9 Spracherkennung

Für die Spracherkennung wird ein Textsample vom Anfang eines Dokuments verwendet. Dieser Text wird in Einzelsätze aufgeteilt und die Sprache für jeden Satz bestimmt. Die erkannte Sprache in der Mehrzahl der Sätze wird zugewiesen. Da linguistische Literatur oft Beispiele in verschiedenen Sprachen enthält werden muss das Textsample hinreichend gross sein. Die Anzahl Zeichen, die verwenden werden sollen kann mit dem Konfigurationsparameter *languageDetectionSampleChars* definiert werden. Es ist zu beachten, dass die Spracherkennung die Laufzeit massgeblich beeinflusst. Deswegen ist es nicht sinnvoll den ganzen Text eines Dokuments dafür zu verwenden. Es werden die N-gram Erkennung (`https://github.com/optimaize/language-detector`) sowie der Spracherkenner (`https://github.com/pemistahl/lingua`) verwendet. Wenn der Konfidenzwert der n-gram Spracherkennung zu niedrig ist wird das Ergebnis des Lingua-Spracherkenners verwendet. Mit dem Konfigurationsparameter *ngramDetectorMinConfidence* wird der Konfidenzwert [0.0,1.0] festgelegt. Mit einem Konfidenzwert > 1.0 lässt sich die n-gram Erkennung abschalten. Dann wird immer die Lingua-Spracherkennung verwendet. Die Erkennung ist für beide Verfahren (n-gram, Lingua) auf ca. 70 Sprachen limitiert (siehe Beschreibung github-repos oben).

2 Datenbank

Extrahierte Metadaten werden zentral in einer Datei gespeichert. Es handelt sich hier nicht um eine Datenbank sondern um eine einfache Serialisierung der Metadatenobjekte. Es stehen Update, Diff und Export Funktionen zur Verfügung. Die Diff-Funktion kann man für die Optimierung der Extraktionsparameter verwenden, indem man beobachtet, ob sich die Qualität der Metadaten verbessert oder verschlechtert, wenn man Parameter verändert. Zum Editieren der extrahierten Metadaten ist die Serialisierung allerdings nicht geeignet.

2.1 Datenbankfunktionen

Die folgenden Funktionen können auf der Kommandozeile oder in der Konfigurationsdatei angegeben werden (siehe dazu Abschnitt 3.2)

- GET_STATS eine Statistik eines Durchlaufs zeigen
- GET_NEW neu gefundene Metadaten anzeigen
- GET_CHANGED veränderte Metadaten anzeigen
- GET_MISSING fehlende Metadaten anzeigen
- COMMIT_NEW neue Metadaten in die Datenbank comitten
- COMMIT_CHANGED geänderte Metadaten abspeichern
- COMMIT_ALL alle gefundenen Metadaten abspeichern
- EXPORT_DB Alle Metadaten in der Datenbank in eine MODS Datei exportieren

2.2 Verwendung der Datenbankfunktionen

Das Erstellen einer Extraktionskonfiguration ist ein iterativer Prozess. Die GET-Befehle lassen sich dabei dazu verwenden um die Qualität der extrahierten Metadaten zu beurteilen. Typischerweise beginnt man damit eine initiale Extraktionskonfiguration zu erstellen. GET_STATS zeigt die einzelnen geparsten Metadaten auf der Kommandozeile und exportiert ausserdem die Ergebnisse als MODS in das jeweilige Datenverzeichnis. Diese werden bei jedem neuen Durchlauf überschrieben. GET_STATS vergleicht ausserdem die extrahierten Metadaten mit schon vorhandenen Metadaten von einem Dokument in der Datenbank, falls dieses schon einmal geparst wurde, und zeigt die alten und neuen Ergebnisse auf der Komanndozeile. Alle anderen GET-Befehle haben dieselbe Funktionalität wie GET_STATS, beschränken der Vergleich jedoch nur auf neue, geänderte oder fehlende Einträge in der Datenbank.
Im nächsten Schritt lassen sich die Extraktionsergebnisse mittels der COMMIT-Operation in die Datenbank schreiben. Diese überschreibt schon vorhandene Metadaten eines Dokument. Mit der EXPORT_DB Operation lassen sich alle Metadaten aus der Datenbank in eine einzelne MODS-Datei exportieren.

3 Kommandozeilen Interface und Konfiguration

3.1 Start der Metadatenextraktion

java -jar mde.jar –config config.json [-mode mode]

-config	Konfigurationsdatei filename.json
-mode	GET_STATS (default) | COMMIT_NEW / CHANDED / MISSING | GET_NEW / CHANDED / MISSING | EXPORT_DB

3.2 Ordnerstruktur für Datenquellen

```
- Root-Ordner  (documentRootDir, runAllSources=true|false)
  - Ordner für Datenquelle 1 (active=true|false)
    - items.mods  (Crawler Metadaten für enthaltene PDFs)
    - PDF/HTML/BIB Dateien
    - Ordner für konvertierte XML-Daten (xmlDir)
      - eventuell Unterordner (mit gleichem Aufbau)
  - ...
  - Ordner für Datenquelle n
```

Beim einem Lauf werden alle aktiven Datenquellen (Feld *active=true*), oder falls der globale Parameter *runAllSources* gesetzt ist, alle Quellen geparst, und die Aktion, die mit dem Parameter *mode* angegeben ist, nach dem Lauf ausgeführt. Im Standardmodus GET_STATS wird das Ergebnis des aktuellen Durchlaufs mit dem Inhalt der Datenbank verglichen und eine Statistik über neue/geänderte bzw. fehlende Metadaten angezeigt. Alle Modi ausser COMMIT_NEW/CHANGED/ MISSING verändern den aktuellen Zustand der Datenbank nicht. Die Ergebnisse aus einem Durchlauf lassen sich mit einer COMMIT-Operation in die Datenbank übernehmen, und mit EXPORT_DB alle Metadaten in das MODS-Format exportieren.

3.3 Sortierfunktion für Dateien einer Datenquelle

Die Sortierfunktion steht unabhängig von der Metadatenextraktion zur Verfügung, und dient der Vorbereitung der Dateien einer Datenquelle für die Verarbeitung. Der Anwendungsfall ist gegeben, wenn in einer Datenquelle verschiedene Ausgaben einer Publikation vorhanden sind, die sich im Layout stark unterscheiden und für die jeweils eine eigene Extraktionsbeschreibung erstellt werden muss. In diesem Fall ist es notwendig für alle gleichformatierten Dateien einen extra Ordner anzulegen. Da das manuelle Sortieren bei vielen Dateien sehr zeitaufwendig ist erledigt ein Algorithmus diese Aufgabe, indem er Dateien mit identischem Namensschema identifiziert. Ein einfaches Beispiel wären z.B. die Dateien einer Zeitschrift, mit folgenden Namensschemata: fel-2018-00.pdf, fel-2018-01.pdf, .. sowie ldd01-01.pdf, ldd01-02.pdf, etc. . Es stehen folgende Parameter für den Clustering-Algorithmus zur Verfügung:

java -jar mde.jar -sort Verzeichnis [-minClusterNameLength n] [-clusterJoinMaxSuffixMissmatch m] [-execute] [-extension ext] [-recursions r]

sort	Sortiert Dateien im Verzeichnis mit ähnlichem Präfix in Unterverzeichnisse
execute	Lege Unterverzeichnisse an und verschiebe Dateien, wenn nicht vorhanden dann gebe nur Cluster aus
recursions	Anzahl der Rekursionen für die Clusterbildung. Für '-1' (default) bricht die Rekursion ab, wenn sich Cluster nicht mehr ändern
extension	Sortiert nur Dateien mir einer bestimmten Dateiendung (Case-insensitiv, default=pdf)
minClusterNameLength	Kürzeste Länge für Clusternamen (default=3)
clusterJoinMaxSuffixMissmatch	Vereinige Clusternamen, die sich in höchstens k Zeichen am Ende unterscheiden (default=6)

3.4 Beschreibung der Konfigurationsdatei config.json

In der Konfigurationsdatei *config.json* stehen :

- am Anfang globale Parameter, wie Dateipfade für die Ein- und Ausgabe
- im Feld *sources* die Extraktionsvorschriften für PDF-Quellen. Die Dateien einer Datenquelle, z.B. einer Zeitschrift sind in einen Ordner, ggf. mit Unterordnern zusammengefasst.

3.5 Globale Konfigurationsparameter

rootDir	Wurzelverzeichnis, Alle Datenquellen befinden sich Unterordnern
xmlDir	Wird automatisch im Unterordner einer Datenquelle für XML-Daten angelegt
databaseDir	Unterverzeichnisname für serialisierte Metadaten (Datenbank)
stopWordFiles	Liste von Dateipfaden mit Begriffen für die NER
affiliationFiles	Liste von Dateipfaden mit Begriffen für Affiliations-Erkennung
stanfordTaggerPath	Pfad von Stanford-Tagger
stanfordNerClassifierPath	Pfad von Stanford NER-Classifier
openNlpPosModelPath	Pfad von Open-NLP POS-Tagger
openNlpNerModelPath	Pfad von Open-NLP NER-Model
runAllSources	Alle Quellen parsen : true | false
runMode	GET_NEW/CHANGED/MISSING/EQUAL| GET_STATS|COMMIT_NEW/CHANGED/ALL| EXPORT_DB
keywordDefinitions	Enthält eine Liste alternativer Schlüsselwort-Definitionen. Eine Schlüsselwortdefinition beinhaltet regex Definitionen für alle Schlüsselworte *title*, *author*, etc. sowie den Parameter *id*. der für die Auswahl in einer Datenquelle mit dem Feld *keywordDefinitionId* verwendet wird.
subTitleSplitRegexRemove	Regulärer Ausdruck mit Trennsymbolen, die für die Subtitelerkennung benutzt werden (wobei das Trennsymbol gelöscht wird, z.B. Bindestrich)
subTitleSplitRegexMaintain	Regulärer Ausdruck, der für die Subtitelerkennung benutzt wird (wobei das Trennsymbol beibehalten wird, z.B. Fragezeichen)
languageDetectionSampleChars	Anzahl Zeichen, die für die Spracherkennung verwendet werden
ngramDetectorMinConfidence	Konfidenz [0.0,1.0] für n-gram Spracherkennung
familyNamePrefixes	Kommagetrennte Liste von Nachnamenspräfixen (z.B. *von*, *van*, *Mc*, die für die Erkennung von Nachnamen verwendet werden)
authorNameFiles	Dateien mit Autorennamen können für die Erkennung von Vor- und Nachnamen benutzt werden. (Parameter: *filePath = Pfad*, *familyNameColumn = Spaltenr. für Nachnamen*, *givenNameColumn = Spaltennr. für Vornamen*)
sources	Enthält die Liste der Extraktionsbeschreibungen für alle Datenquellen

Listing 6: Beispiel : globale Konfigurationsparameter

```
{
"documentRootDir" : "/media/fdata",
"xmlDir" : "xmlData",
"databaseDir" : "serializedData",
"stopWordFiles" : ["/media/fdata/config/ub_stopwords_eng_short",
"/media/fdata/config/lexvo_stopword_university_multiling"
],
"affiliationFiles" : [
"/media/fdata/config/wikidata_q3918_affiliations"
],
"stanfordTaggerPath" : "z.B. english-caseless-left3words-distsim.tagger",
"stanfordNerClassifiersPath" : "z.B. english.all.3class.distsim.crf.ser.gz",
"openNlpPosModelPath" : "z.B. en-pos-maxent.bin",
"openNlpNerModelPath" : "z.B. en-ner-person.bin",
"runAllSources" : false,
"runMode" : "GET_STATS",
"sources" : []
}
```

3.6 Allgemeine Parameter einer Datenquelle

id	Ordnername einer Datenquelle
subId	Optionaler Subordner
active	true | false, falls nicht aktiv wird Datenquelle nicht bearbeitet
keywordDefinitionId	Optionale Schlüsselwortdefinitionen verwenden. Falls nicht angegeben werden die Standarddefinitionen verwendet. Diese sind '.+' für *title*, *journalTitle*, *author*, *date* und *place* sowie '\d+' für *beginPage*, *endPage, year, volume und issue.*
comment	Beliebiger Kommentar
extractorConfig	Enthält die Beschreibung der Extraktionsmethode mit Parametern (optional)

Folgende Verfahren sind kombinierbar:

- Titel, Author(en) und Fusszeilenextraktion
- Zitierangaben und Fusszeilenextraktion
- Inhaltsverzeichnis und Fusszeilenextraktion

3.7 Extraktionsparameter einer Datenquelle im Feld *extractorConfig*

Parameter für Titel und Author Extraktion (Abschnitt 1.2)

bindSequentialLinesBy	fontType | fontSize | fontType+fontSize
firstPage	Seite auf der sich Titel und Author(en) befinden (falls nicht angegeben wird Seite 1 verwendet)

Parameter für die Extraktion von Kopf-/Fusszeilen (Abschnitt 1.4)

footerDescriptions	Liste von Header/Footerbeschreibungen
footerStartsWithText	Regex mit dem der Header/Footer gefunden werden kann
footerDescription	String, der Schlüsselworte (siehe Abschnitt 1.4) und reguläre Ausdrücke enthalten kann
footerPage	Seite auf der sich Header/Footerinformation befindet (falls nicht angegeben wird *firstPage* verwendet)

Parameter für die Extraktion von Zitierangaben (Abschnitt 1.5

citationDescriptions	Liste von Zitiervorgaben
citationStartsAfterText	Regulärer Ausdruck zum Finden des Anfangs einer Zitierangabe
citationEndsBeforeText	Regulärer Ausdruck zum Finden des Endes einer Zitierangabe
citationVariants	Liste von contentDescription
contentDescription	Beschreibung der Zitierinformation (siehe auch Abschnitt 1.5)
firstPage	Die Seite auf der die Zitierangabe steht

Parameter für die Extraktion von Inhaltsverzeichnissen (Abschnitt 1.6)

tocDescriptions	Liste von Beschreibungen für Inhaltsverzeichnisse
tocStartsAfterText	Regulärer Ausdruck zum Finden des Anfangs des Inhaltsverzeichnis
tocEndsBeforeText	Regulärer Ausdruck zum Finden des Endes eines Inhaltsverzeichnis
tocEntryDescription	(siehe Abschnitt 1.6)
contentDescription	(siehe Abschnitt 1.6)
firstPage	Seite auf der das Inhaltsverzeichnis beginnt

4 Verwendete externe Software und Modelle

- pdftohtml version >= 0.67.0 (ist Teil von poppler-utils)
- Stanford POS Tagger (v4.2.0),
  `https://nlp.stanford.edu/software/stanford-tagger-4.2.0.zip`
  `https://nlp.stanford.edu/software/tagger.shtml#Download`
  (nur english-caseless-left3words-distsim.tagger, english.all.3class.distsim.crf.ser.gz)
- Apache OpenNLP - `http://opennlp.sourceforge.net/models-1.5/`
  (nur en-pos-maxent.bin, en-ner-person.bin)
  Mehr Modelle unter `https://opennlp.apache.org/models.html`

Literatur