

Universität Zürich
Institut für Computerlinguistik
Frühlingssemester 2015
Projekt: Bulletin4Corpus
Projektleiter: Prof. Dr. Martin Volk

**Artikel- und Satzalignierung der Dateien des
Credit Suisse Bulletins
(Jahrgänge 1998-2014)**
Programmierprojekt

Dolores Batinić
Dörflistrasse 29
8057 Zürich
dolores.batinic@uzh.ch
Studentin im 12. Semester (MA)
Multilinguale Textanalyse (Monofach)
Matrikelnummer: 09735184

Inhaltsverzeichnis

1 Einleitung und Zielsetzung	3
2 Datenaufbereitung	3
2.1 Konvertierung der PDF-Dateien ins XML Format mit PDF2XML Pipeline	4
2.2 Evaluation des Artikelerkennungsmodells der PDF2XML Pipeline	8
3 Artikelalignierung	16
3.1 Prinzip	16
3.2 Anweisungen	19
4 Satzalignierung	20
5 Fazit	20
6 Literaturverzeichnis	22

1 Einleitung und Zielsetzung

Das Projekt Bulletin4Corpus wurde im Jahr 2015 mit dem Ziel gegründet, ein paralleles Korpus für vier Sprachen (Deutsch, Englisch, Französisch und Italienisch) anhand des Credit Suisse Bulletins (CS Bulletin) zu bauen.¹ Das CS Bulletin wird seit 1895 in gedruckter Version publiziert, wobei seit 1998 auch die PDF-Ausgaben vorhanden sind. In der vorliegenden Arbeit habe ich mich mit den PDF-Ausgaben des CS Bulletins beschäftigt. Das Ziel meines Programmierprojekts war dreifach:

1. Das bestehende Modell für die Erkennung der Artikel aus den PDF Ausgaben des CS Bulletins aufzubereiten und zu evaluieren;
2. Eine TEI-konforme XML-Datei mit den alignierten Artikel aus den parallelen PDF-Ausgaben zu erzeugen (Artikelalignierung); und
3. Parallelle Sätze aus den Ausgaben des CS-Bulletins mit Hilfe der bestehenden Skripten aus dem Text&Berg² Projekt zu verbinden (Satzalignierung).

Das Ausgangsmaterial für die Evaluation des Artikelerkennungsmodells (1) und die anschliessende Artikel- und Satzalignierung (2,3) waren 73 parallelen Ausgaben des CS Bulletins, die 239 PDF-Dateien entsprachen. Davon waren 73 Ausgaben auf Deutsch, 73 auf Französisch, 58 auf Italienisch und 30 auf Englisch.³

Für die Ziele dieses Programmierprojekts habe ich mit Prof. Martin Volk (Satzalignierung) und mit den Kommilitonen Noëmi Aepli (Datenaufbereitung und Evaluation), Till Salinger (Datenaufbereitung) und Fabienne Leuenberger (Satzalignierung) zusammengearbeitet.

2 Datenaufbereitung

Um die Artikel- bzw. Satzalignierung ausführen zu können, mussten alle PDF-Dateien des CS Bulletins in ein XML Format konvertiert werden. Für die Konvertierung der Dateien habe ich die PDF2XML Pipeline von Till Salinger benutzt (Salinger 2015), die speziell für das Projekt Bulletin&Corpus gebaut wurde. Die PDF2XML Pipeline hatte den Anspruch, von den PDF-Dateien des CS Bulletins die XML-Dateien zu produzieren, welche die gleiche Form wie die

¹ bulletin4corpus in: <<http://kitt.cl.uzh.ch/kitt/b4c/en/index.php>>.

² Text+Berg Digital in: <<http://textberg.ch/site/en/welcome/>>

³ Die Ausgaben von 1998/1, 1998/2, 1998/3 und 2002/3 waren nur auf Deutsch verfügbar. Für die Jahrgänge 2008, 2009, 2010 und 2011 war kein Material vorhanden.

parallelen XML-Dateien des Projekts Text&Berg haben.⁴

2.1 Konvertierung der PDF-Dateien ins XML Format mit PDF2XML Pipeline

Um das gleiche Format wie die Text&Berg Dateien zu haben, mussten die XML-Dateien des CS Bulletins einen `article` Tag haben, in dem der Inhalt der Artikel gespeichert ist. Beim Projekt Text&Berg wurde die Markierung der Artikelgrenzen vollkommen manuell gemacht, beim Bulletin&Corpus wurde jedoch versucht, die Markierung der Artikel so automatisiert wie möglich zu machen. Zu diesem Zweck hat Salinger eine Web-Oberfläche vorbereitet, auf welcher die Artikelgrenzen markiert werden können. Wenn alle Artikel erkannt worden sind, können die XML-Dateien mit dem `article` Tag produziert werden, die für eine Artikelalignierung notwendig sind. Für die Satzalignierung müssen die XML-Dateien PoS-getaggt und lemmatisiert werden.

Die Schritte, die ich machen musste, um die PDF-Dateien in das gewünschte XML Format zu bekommen, sind die folgenden:

1. Erstellung der TETML-Dateien (PDF → TETML)
2. Artikelerkennung
 - a) Erstellung der HTML-Dateien (TETML → HTML)
 - b) Erstellung der Parameter-Dateien für die Erkennung der Artikelgrenzen (manuell)
3. Erstellung der *intermediate* XML-Dateien mit `article` Tags mit Hilfe der Parameter-Dateien (Parameter-Dateien + TETML → XML)
4. Erstellung der XML-Dateien im Text&Berg Format (*intermediate* XML→ *Text&Berg* XML)

Im Schritt (1) wurden die PDF-Dateien in das TETML-Format konvertiert. Die Konvertierung erlaubte es, die Information über die Fontnamen und über die Schriftgrösse von jedem Textteil zu erkennen. Dieser Schritt musste auf dem Server

⁴ In den XML-Dateien von Text&Berg wurde inzwischen die Nummerierung der Artikel, Sätze und Wörter von 1-1-1 zu a1-s1-w1 geändert, während die XML-Dateien der PDF2XML Pipeline die alte Nummerierung aufweisen (beides 1-1-1 und a1-s1-w1 steht für das erste Wort im ersten Satz vom ersten Artikel).

hal2 ausgeführt werden, weil dort alle nötigen Libraries für die Konvertierung installiert worden sind.

Im Schritt (2) mussten zuerst die HTML-Dateien erzeugt werden, in denen die Textteile anhand von den Fontnamen und Schriftgrösse getrennt wurden (Schritt 2.a). Nach der Erstellung aller HTML-Dateien konnte die Markierung der Überschriften auf der Web-Oberfläche beginnen (Schritt 2.b). Die Web-Oberfläche ohne markierte Überschriften sah wir in der Abbildung 1 aus.

Text Snippet	Font Name	Font Size
N° 5	AkzidenzGroteskBE	11
/	AkzidenzGroteskBE	11.5
2014	AkzidenzGroteskBE	11
Das älteste Bankmagazin der Welt. Seit	ACaslonPro	16
1895	ACaslonExp	16
.	ACaslonPro	16
Frage 1 —	ACaslonPro	13
Ist die Schweiz das Land der Berge?	ACaslonPro	20
25 Fragen zur Schweiz	ACaslonPro	38
Und: Credit Suisse Sorgenbarometer – die grosse Studie zur Lage der Nation	ACaslonPro	16

The right side shows a magazine cover titled "Bulletin" from 2014. The cover features a woman wearing a hat and glasses, with the Matterhorn in the background. Text on the cover includes "Das älteste Bankmagazin der Welt. Seit 1895.", "Ist die Schweiz das Land der Berge?", "25 Fragen zur Schweiz", and "Und: Credit Suisse Sorgenbarometer – die grosse Studie zur Lage der Nation".

Abbildung 1: Web-Oberfläche für die Markierung der Titel anhand der Fontnamen und Schriftgrößen, CS Bulletin 2014/5/de

Auf der Web-Oberfläche wurde für jede HTML-Datei die entsprechende PDF-Datei angehängt. Für die Markierung der Artikelgrenzen musste ich die Überschriften der Artikel im PDF finden und auf die entsprechenden Fontnamen/Schriftgrößen klicken. Jeder neue Klick hat einen Eintrag in der Parameter-Datei *Headings.xml* erzeugt (vgl. Abbildung 3). Um die Markierung der Überschriften zu beschleunigen, mussten die Textteilen, deren Schriftgröße kleiner als 20 war, rausgefiltert werden (Einstellung: *Min font size: 20/Higlight Font Size/Filter table*). Nach der Filtrierung wurden alle Überschriften, die grösser als 20 waren, blau gefärbt und klickbar gemacht, d.h. als möglichen Überschriften vorgeschlagen. Wenn eine Überschrift als solche ausgewählt wurde, änderte sich der Fontname bzw. die Schriftgröße

zu grün (vgl. Abbildung 2), wobei auch alle anderen Textteile mit den gleichen Fontnamen/Schriftgrößen automatisch grün markiert wurden.

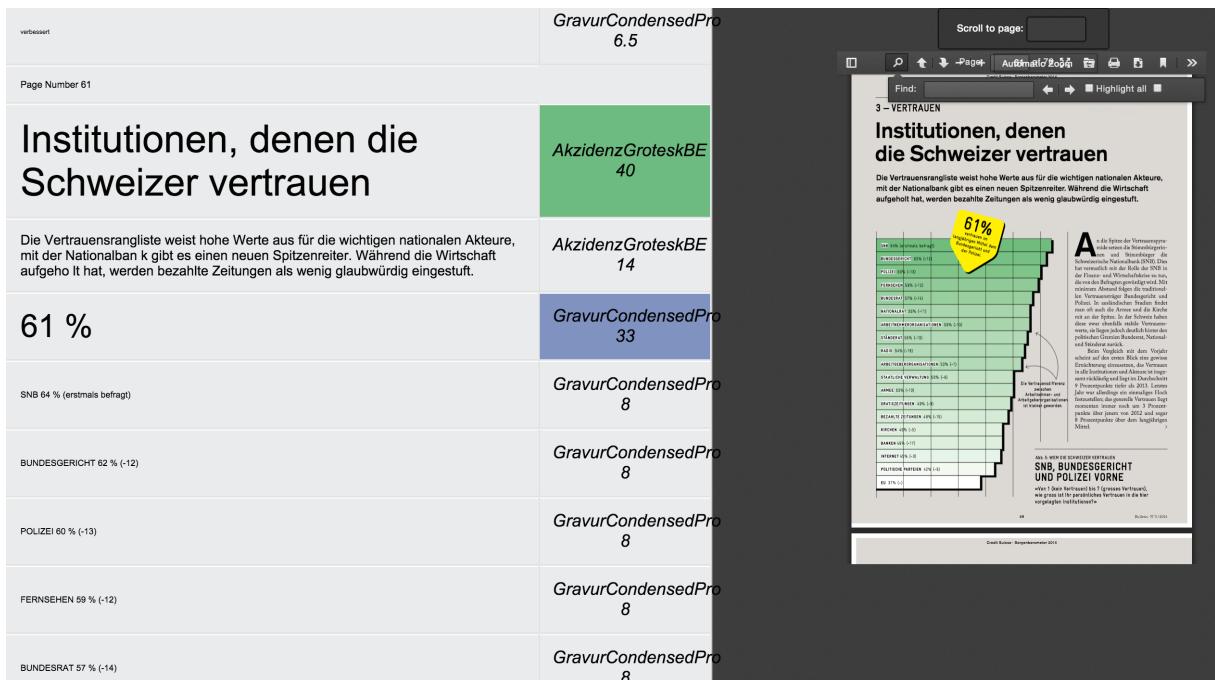


Abbildung 2: Markierte Überschrift (grün), CS Bulletin 2014/5/de

Diese Einstellung war einerseits sehr hilfreich, weil mehrere Überschriften auf einmal markiert werden konnten und andererseits ungünstig, weil auch die Textteile, die keine Überschriften waren, als solche gekennzeichnet wurden. Dabei war es nicht möglich, die falsche Markierung(en) rückgängig zu machen, ohne die markierten Überschriften zu löschen. Die Unmöglichkeit False Positives zu entfernen ohne True Positives miteinzubeziehen hat zusammen mit einigen Formattierungsbesonderheiten die Erkennung der Artikelgrenzen und die anschliessende Artikelalignierung wesentlich erschwert (vgl. Abschnitt 2.2 und 3). Um den Markierungsprozess weiter zu beschleunigen, gab es noch die Möglichkeit, die Fontnamen und/oder die Schriftgrößen, die in den schon bearbeiteten Dateien ausgewählt wurden, klickbar zu machen (Einstellung: *Choose file*). Insgesamt hatte Salinger die Zeit für die Markierung aller Überschriften in einer Ausgabe auf zwei Minuten geschätzt, was für 239 Dateien 478 Minuten (8 Stunden) beanspruchen würde.

Die Parameter-Dateien für die Jahrgänge 2012, 2013 und 2014 (66 Ausgaben) hat Noëmi Aepli generiert, während die restlichen Jahrgänge (174 Ausgaben) von mir

erzeugt wurden. Ein Beispiel für die Parameter-Datei ist in der Abbildung 3 gezeigt.

```
<!DOCTYPE BULLETIN [  
  <!ELEMENT BULLETIN (HEADING)+>  
  <!ELEMENT HEADING EMPTY>  
  <!ATTLIST HEADING name CDATA #REQUIRED>  
  <!ATTLIST HEADING fontabbrv CDATA #REQUIRED>  
  <!ATTLIST HEADING fontsize CDATA #REQUIRED>  
>  
<BULLETIN>  
  <HEADING name="article" fontabbrv="ACaslonPro" fontsize="22"/>  
  <HEADING name="article" fontabbrv="ACaslonPro" fontsize="28"/>  
  <HEADING name="article" fontabbrv="ACaslonPro" fontsize="30"/>  
  <HEADING name="article" fontabbrv="ACaslonPro" fontsize="33"/>  
  <HEADING name="article" fontabbrv="ACaslonPro" fontsize="38"/>  
  <HEADING name="article" fontabbrv="ACaslonPro" fontsize="42"/>  
  <HEADING name="article" fontabbrv="ACaslonPro" fontsize="45"/>  
  <HEADING name="article" fontabbrv="ACaslonPro" fontsize="50"/>  
  <HEADING name="article" fontabbrv="AkzidenzGroteskBE" fontsize="40"/>  
  <HEADING name="article" fontabbrv="AkzidenzGroteskBE" fontsize="50"/>  
  <HEADING name="article" fontabbrv="GravurCondensedPro" fontsize="40"/>  
  <HEADING name="article" fontabbrv="Quatro" fontsize="33.41"/>  
</BULLETIN>
```

Abbildung 3: Parameter-Datei für CS Bulletin 2014/5/de

Sobald für jede der 239 Ausgaben des CS Bulletins eine Parameter-Datei vorhanden war, konnten die *intermediate* XML-Dateien generiert werden (Schritt 3). In den *intermediate* XML-Dateien wurden die Text-Abschnitte aus den TETML-Format übernommen, wobei die `article` Tags an den entsprechenden Stellen im XML hinzugefügt wurden, welche den Fontnamen und Schriftgrößen aus den Parameter-Dateien entsprachen. Alle `article` wurden mit dem XML Attribut `@n` nummeriert, wobei es für jede Ausgabe auch einen `article n="0"` gab, in dem die Metainfomationen und alles, was vor den Artikeln vorkam, gespeichert wurde (Titelseite, Inhaltsverzeichnis, usw.).

Im Schritt 4 wurden die *intermediate* XML-Dateien dem Text&Berg Format angepasst, d.h. sie wurden lemmatisiert und PoS-getaggt, wobei jedem Satz ein `s` Tag und jedem Wort ein `w` Tag zugewiesen wurden. Dieser Schritt musste wie Schritt 1 auf dem Server `hal2` ausgeführt werden. Die Konvertierung der *intermediate* XML-Dateien in das Text&Berg Format hat die Satzalignierung ermöglicht, währenddessen für die Artikelalignierung die *intermediate* XML-Dateien gereicht haben. Die PDF2XML Pipeline ist auf `hal2` in im Ordner `/home/user/salinger` gespeichert. Die Anweisungen für die Ausführung der Skripte aus der PDF2XML Pipeline

sind im Anhang 1 zu finden.

2.2 Evaluation des Artikelerkennungsmodells der PDF2XML Pipeline

Vor der Artikelalignierung habe ich eine Evaluation durchgeführt, um zu sehen, inwiefern die PDF2XML Pipeline für die Erkennung der Artikel weiter verwendet werden kann.

Für die Analyse des Artikelerkennungsmodells habe ich folgende Kriterien berücksichtigt:

1. Übereinstimmungen zwischen geklickten Überschriften und Anzahl Artikel im XML; und
2. Übereinstimmungen zwischen der Anzahl Artikel im PDF und Anzahl Artikel im XML.

Bei dem Punkt 1. habe ich analysiert, ob die Anzahl markierter Überschriften mit der Anzahl der `article` Tags im XML übereinstimmt und beim Punkt 2. habe ich die tatsächliche Anzahl Artikel (Anzahl Artikel im Inhaltsverzeichnis der Ausgaben im PDF Format) mit der Anzahl der `article` Tags in den jeweiligen XML-Dateien verglichen. Die Anzahl markierter Überschriften wurde auf der Web-Oberfläche am Ende der Verarbeitung jeder Ausgabe angezeigt. Für das Zählen der Artikel in den XML-Dateien (`article` Tags) habe ich den Skript `CountArticleTagsInXML.sh` geschrieben (Anhang 1), während die Anzahl Artikel in den PDF-Dateien für jede Datei manuell gezählt wurde (vgl. Anhang 1). Die Auszählung habe ich zusammen mit Noëmi Aepli gemacht, wobei sich Aepli wie bei der Erstellung der Parameter-Dateien mit den Jahrgängen 2012, 2013 und 2014 (66 Ausgaben) beschäftigte und ich für die restlichen Jahrgänge (174 Ausgaben) verantwortlich war.

Die Tabelle mit den vollständigen Resultaten der Evaluation für insgesamt 239 Ausgaben des CS Bulletins ist im Anhang 1 illustriert, während die Anzahl der Übereinstimmungen der Punkte 1 und 2 in der Tabelle 1 aufgeführt ist.

Tabelle 1: Übereinstimmungen

	Anzahl Klicks vs. Anzahl Artikel im XML	Anzahl Artikel im PDF vs. Anzahl Artikel im XML
Übereinstimmungen	228	15
Keine Übereinstimmungen	11	224
Total	239	239

Wie in der Tabelle 1 ersichtlich, stimmt die Anzahl markierten Überschriften mit der Anzahl der `article` Tags im XML in 95,4% Fälle überein (228/239), während die Anzahl Artikel im PDF nur in 0.06% der Fälle (15/239) übereinstimmt.⁵ Die Diskrepanz zwischen der Anzahl Artikel im XML und im PDF ist auch im Diagramm 4 zu sehen.

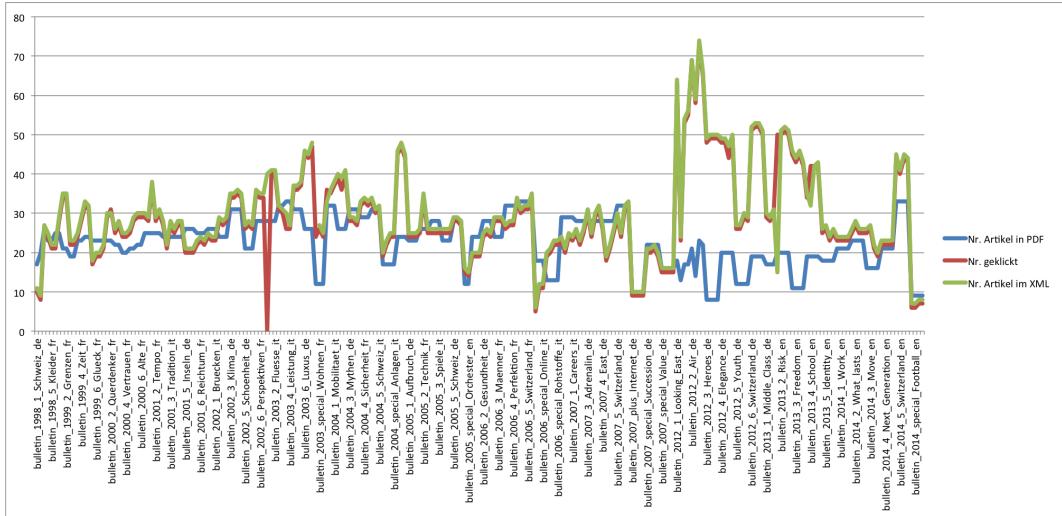


Abbildung 4: Übereinstimmungen

Wie im Diagramm 4 gezeigt, hat das Artikelerkennungsmodell der PDF2XML Pipeline die Überschriften, die auf der Web-Oberfläche angeklickt worden sind (rot), in den meisten Fällen, wie erwartet, als Artikelgrenzen in den XML-Dateien erkannt (grün). Wo das Modell aber gescheitert ist, liegt an der Grundaufnahme, dass die Artikel-Überschriften anhand von ihren Fontanamen und Schriftgrößen erkannt werden können und dass sie weiter für die Markierung der Artikelgrenzen benutzt werden können. Diese Annahme hat sich, wie im Abschnitt 2 bereits erwähnt, als nicht ganz zutreffend erwiesen. Die Gründe hierfür sind die folgenden:

1. Größere Schriftgrößen sind nicht nur für die Artikel-Überschriften reserviert: auch andere Textteile (Abbildung 5), Illustrationen im Hintergrund (Abbildung 6) oder die Werbung der Credit Suisse (Abbildung 7) sind teilweise gleich gross wie die Überschriften geschrieben.
2. Eine Überschrift kann unterschiedliche Schriftgrößen haben.

⁵ Es galt als Übereinstimmung, wenn die Anzahl Artikel im XML +1 grösser als die Anzahl Artikel im PDF bzw. Anzahl markierten Überschriften war, da im XML ein zusätzlicher Artikel `article n="0"` hinzugefügt wurde.

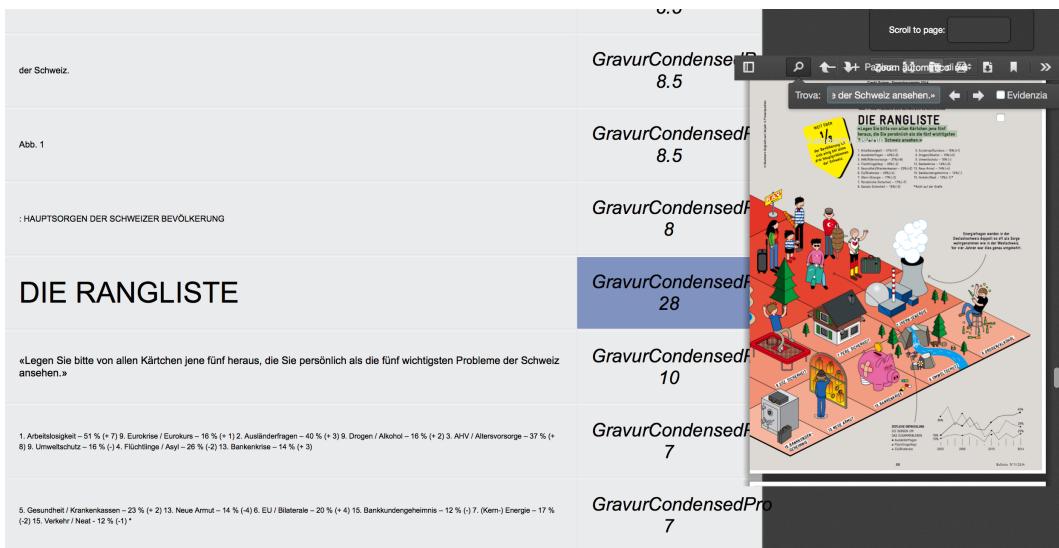


Abbildung 5: Grosse Schriftgrösse obwohl es sich nicht um einen Titel handelt, CS Bulletin 2014/5/de

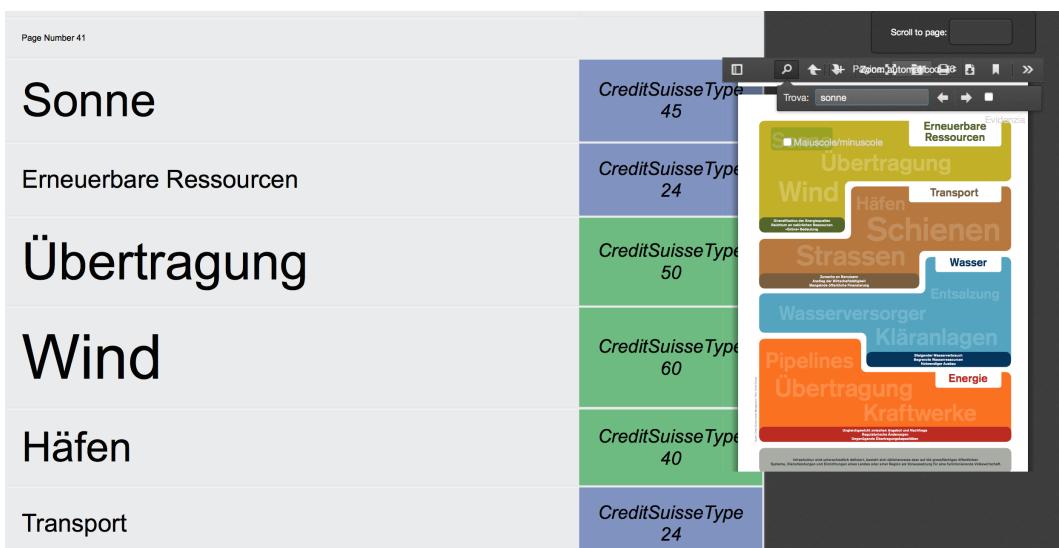


Abbildung 6: Illustrationen im Hintergrund als Überschriften markiert, CS Bulletin 2012/2/de

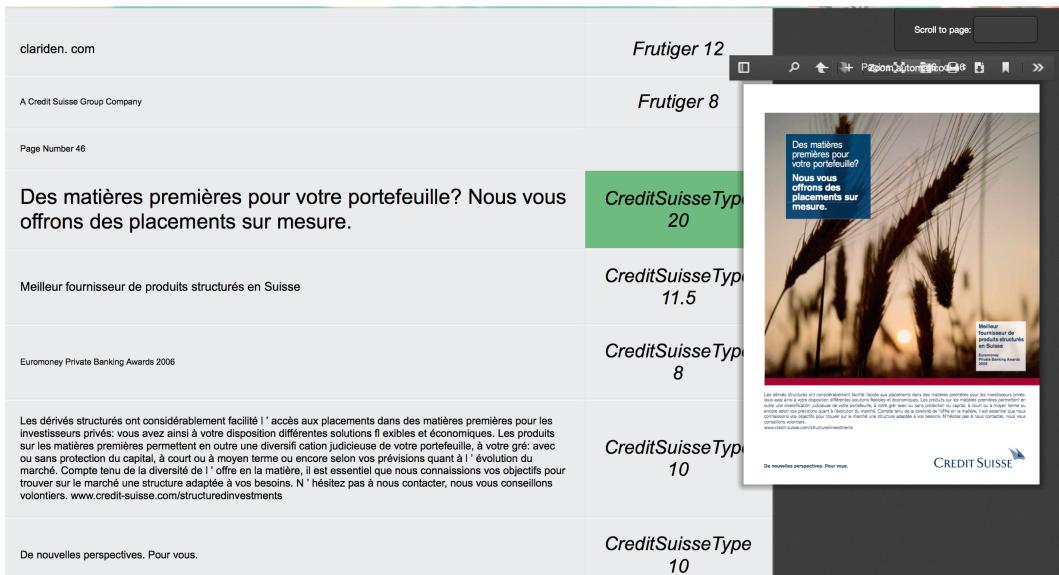


Abbildung 7: Werbung der Credit Suisse als Überschrift markiert, CS Bulletin 2006/spezial:Rohstoffe/fr

Dem Artikelerkennungsmodell schien es vor allem an Präzision zu fehlen; wegen der automatischen Markierung von False Positives, ergab das Modell in 65% der Fälle mehr `article` Tags im XML, als dass es Artikel im PDF gibt (vgl. Anhang 1: in 156 von 239 XML-Dateien hat das Modell mehr Artikel als in den entsprechenden PDF-Dateien erkannt). Die Frage der Präzision war vor allem deswegen wichtig, weil der Artikelerkennung eine Artikelalignierung folgten sollte.

Weitere Herausforderungen, die die PDF2XML Pipeline im Bezug auf die Artikelerkennung vorgewiesen hat, fanden bei der Konvertierung von PDF zu TETML und/oder von TETML zu HTML statt, wie zum Beispiel die Auflösung der Abstände zwischen Wörtern (Abbildung 8) oder die Kodierungsprobleme (Abbildung 9).

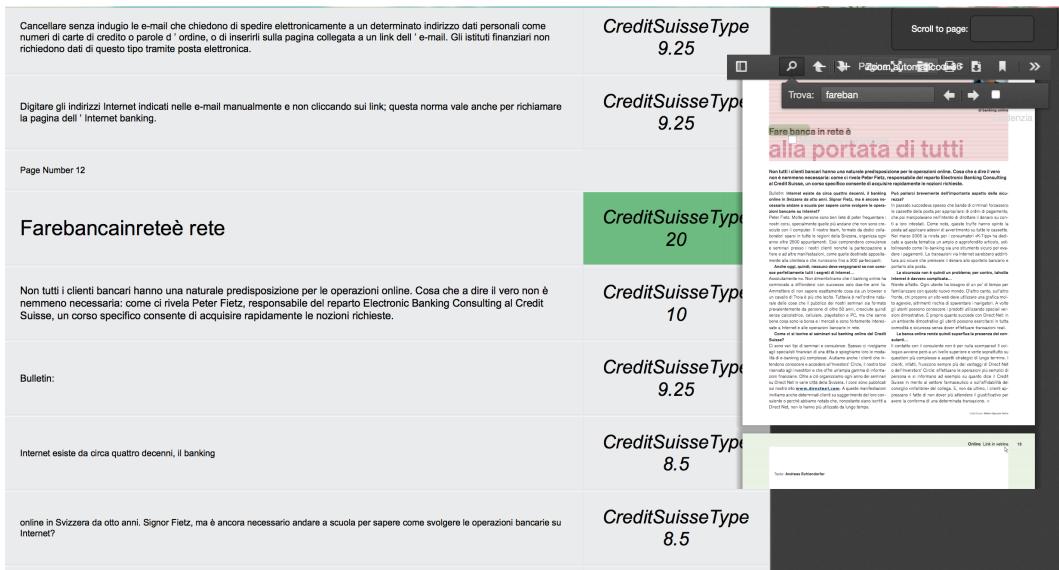


Abbildung 8: Keine Abstände im Titel, CS Bulletin 2006/special:Online/it

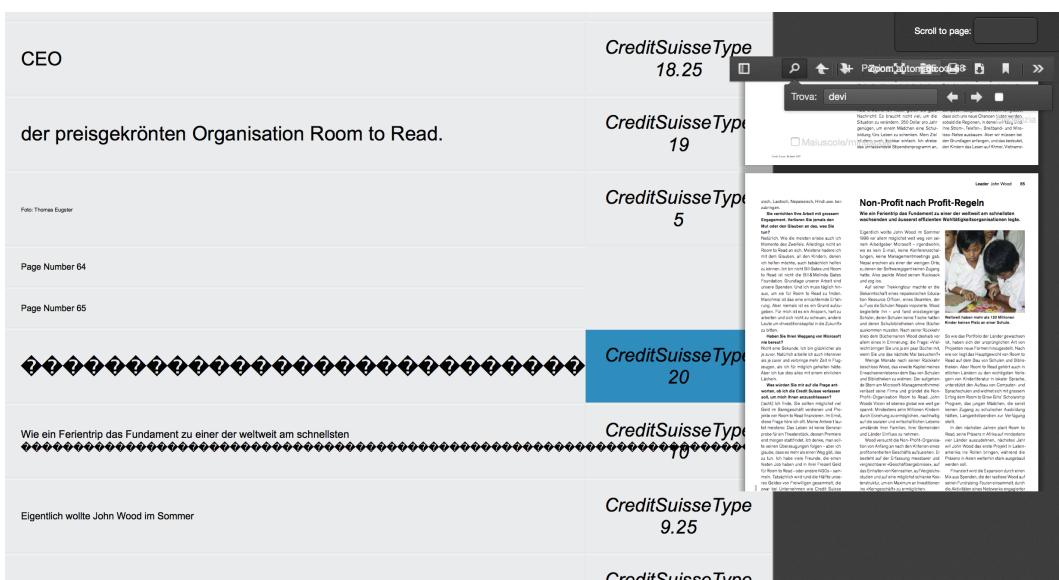


Abbildung 9: Kodierung-Probleme, CS Bulletin 2007/1/de

Was die Funktionalitäten und die Benutzerfreundlichkeit angeht, kann folgendes gesagt werden: Die Web-Oberfläche, auf der die Überschriften markiert wurden, hat sich als eine sehr nützliche Plattform für die Erstellung der Parameter-Dateien erwiesen. Am anschaulichsten war die Web-Oberfläche mit Chrome Browser, weil die anderen Browser die Einstellungen für die Darstellung der Schriftgrösse auf dem Bildschirm zum Teil verändert haben. Die Filterung der Schriften, deren Schriftgrösse kleiner als 20 war, hat sich ebenfalls als sehr hilfreich erwiesen, obwohl es möglicherweise Überschriften gab, die kleiner geschrieben waren.

Die durchschnittliche Zeit, die ich für die Markierung der Überschriften investiert habe, war zwei bis drei Minuten pro Ausgabe (abhängig von der Anzahl der Artikel).

Zusammenfassend kann man sagen, dass das Artikelerkennungsmodell von der PDF2XML Pipeline für die Erkennung der Artikelgrenzen im CS Bulletin nicht gut geeignet ist, vor allem weil die Schriftgrößen und Fonts in der ursprünglichen PDF-Dateien nicht konsistent sind. Die Güte des Artikelerkennungsmodell von der PDF2XML Pipeline wäre deswegen anhand weiterer Zeitschriften zu testen, die eine einheitliche Formatierung vorweisen.

In einer weiteren Implementierung des Artikelerkennungsmodells, empfehle ich mit einem anderen Ansatz zu arbeiten. Ich würde die Entwicklung eines Modells vorschlagen, welches die Artikel anhand von den PDF-Seiten, auf welchen sie anfangen, erkennt (eine PDF-Seite fängt schon mit dem Titelblatt an, während die Nummerierung der Seiten in der Zeitschrift nicht unbedingt vom Titelblatt anfängt, vgl. Abbildung 10, wo die PDF-Seite 4 der Zeitschrift-Seite 2 entspricht, da die Nummerierung der Seiten erst ab der PDF-Seite 2 anfängt). Der Ansatz des Artikelerkennungsmodells von der PDF2XML Pipeline würde ich vielleicht nur dann nutzen, um anhand der erkannten Überschriften zwischen Artikel zu unterscheiden, die auf der gleichen Seite anfangen.

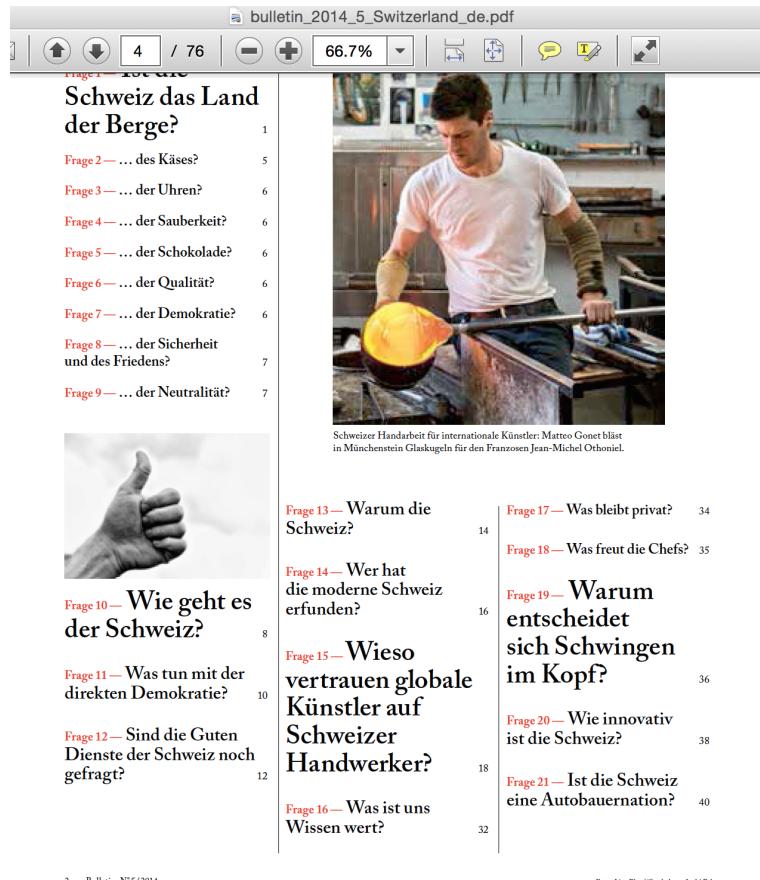


Abbildung 10: Illustratives Beispiel der Inkongruenz zwischen einer PDF-Seite und einer Zeitschrift-Seite, CS Bulletin 2014/5/de

Für eine Weiterentwicklung des Artikelerkennungsmodells, schlage ich den folgenden Ansatz vor:

1. PDF zu XML konvertieren;
2. Abstand zwischen PDF-Seite und Zeitschrift-Seiten messen;
3. Die Zeitschrift-Seiten finden, an welchen das Inhaltsverzeichnis verweist;
4. Den Abstand der Zeitschrift-Seiten, an welchen die jeweiligen Artikel anfangen, mit den Zeitschrift-Seiten addieren, um die PDF-Seiten, auf welchen die Artikel anfangen, zu finden; und
5. Jede PDF-Seite, auf der ein Artikel anfängt, als Artikelgrenze markieren (Annahme: Artikel fangen am Anfang der Seite an).

Der vorgeschlagene Ansatz würde alle Artikel erkennen, wenn es pro Seite nur einen Artikel gibt. Wenn allerdings mehrere Artikel auf der gleichen Seite anfangen, wie

zum Beispiel in der Abbildung 11, müsste zwischen einzelnen Artikel unterschieden werden. In diesem Fall wäre die Information über die Schriftgrösse zwischen Überschriften vielleicht doch nützlich, um die manuelle Arbeit zu vermeiden. Es sollte aber zuerst überprüft werden, ob die Anzahl solcher Fälle genug gross ist, um ein System für die Erkennung der Überschriften entwickeln zu müssen, oder ob eine manuelle Korrektur der Einzelfälle doch genügen würde.

The screenshot shows a PDF document titled "bulletin_1999_6_Glueck_de.pdf". The page number is 38. The main content area has a header "AKTUELL" and features several small articles:

- WWW.CSPB.COM: ONLINE BERATUNG, ONLINE INFORMATION** (with a red box around it)
- CREDIT SUISSE HOLT WIEDER GOLD** (with a gold medal image)
- VOM GÖTTIBATZEN ZUM STARTKAPITAL** (with a hand holding a string and a small gift bag)
- SICHER DURCH DEN BVG-DSCHUNGEL** (with a blue and white graphic)
- GESCHENK-SPARKONTO** (with a text block about it)

At the bottom, there is a footer with page number 36 and the text "CREDIT SUISSE BULLETIN 6/99".

Abbildung 11: Mehrere kleine Artikel auf einer Seite, CS Bulletin 1999/6/de

3 Artikelalignierung

Die Artikelalignierung habe ich anhand von den *intermediate* XML-Dateien gemacht. Da das Artikelerkennungsmodell von der PDF2XML Pipeline in 94% der Fälle mit der Anzahl der Artikel im PDF nicht übereinstimmte und da die Anzahl Artikel in den parallelen Dateien oft nicht übereinstimmte (vgl. Anhang 1) habe ich entschieden, die Artikel anhand der PDF-Seiten zu alignieren. Die Alignierungsstrategie habe ich gewählt, nachdem ich für 10 Dateien aus unterschiedlichen Jahrgängen überprüft habe, dass die Anzahl PDF Seiten gleich ist und dass parallele Artikel auf denselben PDF-Seiten anfangen. Unterschiede in PDF-Seiten Anzahl zwischen parallelen Dateien konnte ich ausschliesslich bei den englischen Ausgaben aus dem Jahr 2007 sowie bei zwei englischen Ausgaben aus dem Jahr 2012 feststellen⁶. Den fehlenden englischen Artikel wurde bei der Artikelalignierung der Wert “None” zugewiesen, damit die Reihenfolge der Artikel konsistent blieb.

3.1 Prinzip

Das Prinzip für die Artikelalignierung war das folgende:

- Aligniere nur die Artikel (`article` Tags), die die gleiche PDF-Nummer (`pb` Tags) haben.

Mit der Alignierung anhand von der PDF-Nummer (`pb` Tags) konnten die fehlerhaften Alignierungen, die wegen der unpräzisen Artikelerkennung verursacht werden könnten, vermieden werden, da die PDF-Nummer ein zuverlässiges Hinweis für die Artikelgrenzen war.

Ein Beispiel vom Alignierungsprinzip ist in der Abbildung 12 X illustriert, wo die zwei Spalten auf zwei parallelen Ausgaben (de/en) stehen und die Alignierung mit einem roten Pfeil gekennzeichnet ist. Die XML-Datei, in der die Alignierungen gespeichert sind, ist TEI-konform⁷

⁶ Folgende englische Ausgaben hatten eine ungleiche Anzahl PDF-Seiten: 2007/1, 2007/2, 2007/3, 2007/4, 2007/5, 2007/special:Succession, 2012/1 und 2012/2

⁷ TEI (Text Encoding Initiative) in: <www.tei-c.org>.

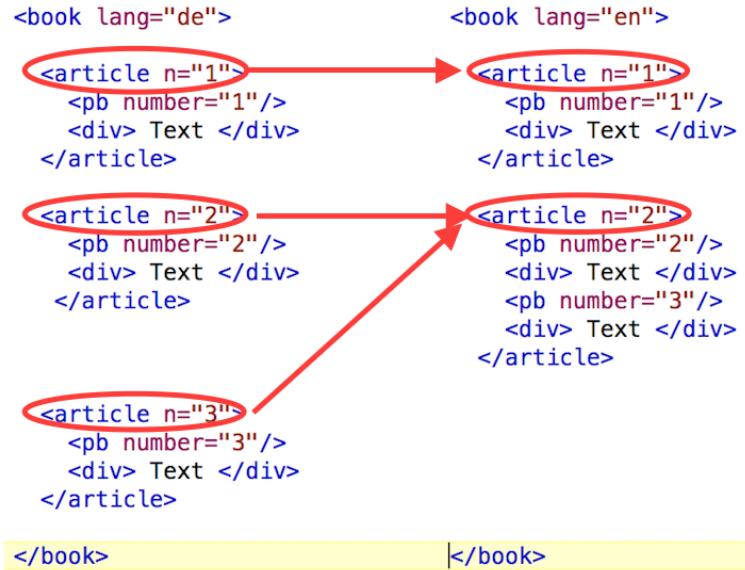


Abbildung 12: Alignierungsprinzip

Als Eingabe für den Artikelalignierungsskript dienten 249 XML-Dateien stammend von 73 parallelen Ausgaben. Für jede Ausgabe wurde der Titel der Ausgabe `teiHeader`, Erscheinungssprache der Ausgaben (`linkGrp[@lang]`) sowie die Dateinamen (`linkGrp[@xtargets]`) gespeichert. In `link` Tags wurden die Artikelalignierungen gespeichert, wobei die Ordnung der Alignierung der Ordnung der Sprachen in `linkGrp[@lang]` entsprach. Die Output-XML-Datei wurde nach dem Muster wie in der Abbildung 13 gebaut, wobei für jede parallele Ausgabe die `teiHeader` und `linkGrp` standen.

```

<TEI>
  <teiHeader>bulletin_JJJJ_Nr_Name</teiHeader>
  <linkGrp lang="lang1;lang2" targType="yearbook" xtargets="bulletin_JJJJ_Nr_Name_lang1.xml;bulletin_JJJJ_Nr_Name_lang2.xml">
    <link targType="article" xtargets="a1;a1"/>
    <link targType="article" xtargets="a2;a2"/>
    <link targType="article" xtargets="a3;a3"/>
    <link targType="article" xtargets="a4;a4"/>
    <link targType="article" xtargets="a5;a5"/>
  </linkGrp>
</TEI>

```

Abbildung 13: Beispiel vom TEI-Output: Alignierung von zwei parallelen Dateien in Sprache A (lang1) und Sprache B (lang2) mit je fünf Artikel

Wie schon erwähnt, konnten die Artikel nicht eins-zu-eins aligniert werden, weil die Anzahl Artikel in den parallelen XMLDateien oft nicht übereinstimmte. Wenn ein Artikel in Sprache A zu mehreren Artikeln in Sprache B entsprach, habe ich die Alignierungen (Werte im `xtargets` Attribut vom Tag `link`) wie in Abbildung 14 gespeichert. Die deutsche XML-Datei des CS Bulletins 2006/spezial:Online hatte nur fünf erkannten Artikel, während die XML-Dateien in Französisch und Italienisch je

11 erkannte Artikel hatten. Die Anzahl `link` Tags entsprach der Anzahl Artikel von der Ausgabe in der Sprache, die am meisten Artikel hatte. Eine andere Möglichkeit für die Struktur der Alignierungen wäre gewesen, die Struktur von der Satzalignierungen vom Text&Berg Korpus zu übernehmen. Dies hätte zur Folge gehabt, dass die Ausgabe der Artikelalignierung wie in Abbildung 15 ausgesehen hätte und dass die Anzahl `link` Tags der Anzahl Artikel der Ausgabe in der Sprache entsprochen hätte, die am wenigsten Artikel hatte. Obwohl die zweite Variante eleganter erscheint, habe ich sie nicht verwendet, weil die Skripte für die Satzalignierung vom Text&Berg Projekt dieses Format der Artikelalignierungen nicht unterstützen.

```
<teiHeader>bulletin_2006_special_Online</teiHeader>
<linkGrp lang="de;fr;it" targType="yearbook" xtargets="t"
    <link targType="article" xtargets="a0;a0;a0"/>
    <link targType="article" xtargets="a1;a1;a1"/>
    <link targType="article" xtargets="a2;a2;a2"/>
    <link targType="article" xtargets="a2;a4;a3"/>
    <link targType="article" xtargets="a2;a5;a4"/>
    <link targType="article" xtargets="a2;a6;a5"/>
    <link targType="article" xtargets="a2;a7;a6"/>
    <link targType="article" xtargets="a3;a8;a7"/>
    <link targType="article" xtargets="a4;a9;a8"/>
    <link targType="article" xtargets="a5;a10;a9"/>
    <link targType="article" xtargets="a5;a11;a11"/>
</linkGrp>
```

Abbildung 14: Beispiel vom TEI-Output: Alignierungen mehr-zu-eins, CS Bulletin 2006/spezial:Online

```
<teiHeader>bulletin_2006_special_Online</teiHeader>
<linkGrp lang="de;fr;it" targType="yearbook" xtargets="bulletin_2006_special_Online_de.x"
    <link targType="article" type="1:1:1" xtargets="a0;a0;a0"/>
    <link targType="article" type="1:1:1" xtargets="a1;a1;a1"/>
    <link targType="article" type="1:5:5" xtargets="a2;a2,a4,a5,a6,a7;a2,a3,a4,a5,a6"/>
    <link targType="article" type="1:1:1" xtargets="a3;a8;a7"/>
    <link targType="article" type="1:1:1" xtargets="a4;a9;a8"/>
    <link targType="article" type="1:2:2" xtargets="a5;a10,a11;a9,a11"/>
</linkGrp>
```

Abbildung 15: Zweite Möglichkeit der Alignierungen mehr-zu-eins, CS Bulletin 2006/spezial:Online

Die Artikelalignierung anhand der PDF-Tags konnte die Folgen der fehlerhaften Artikelerkennung beheben: Wenn eine Überschrift mit zwei unterschiedlichen Schriftgrößen geschrieben wurde, produzierte diese nach dem Artikelerkennungsmodell aus der PDF2XML Pipeline zwei unterschiedliche `article` Tags, wobei die ersten

Artikel `article` Tags leer waren. Die leeren Artikel beinhalteten keine Information zu der PDF-Seitenzahl und darum wurden sie nach dem Alignierungsprinzip richtigerweise nicht aligniert. Es wurden nur diejenigen Artikel aligniert, die einen Inhalt hatten, unabhängig davon, was ihre Artikelnummer (`article[@n]`) ist (vgl. Abbildung 16, wo im CS Bulletin 1999/fr/2 `article n="12"` leer ist).

```
<!--CS Bulletin 1999/2/fr-->

<!--...-->
<article n="12">
  <div heading="article"> «MENSCHEN EN GRENZEN»</div>
  <div> CHRISTIAN PFISTER</div>
  <div> Sind Grenzen ein Hindernis für den Frieden ,</div>
  <div> CORNELIO SOMMARUGA , PRÄSIDENT DES INTERNATIONALEN KOMITEES VOM ROTEN KREUZ</div>
  <div> INTERVIEW : CHRISTIAN PFISTER , REDAKTION BULLETIN</div>
  <div> oder machen sie Frieden gar erst möglich ?</div>
  <div> CORNELIO SOMMARUGA</div>
  <div> Sowohl das eine wie das andere trifft zu .</div>
  <div> Europa beispielsweise hat sich entwickelt , indem sich auf seinem Territorium imme
  <div> 23</div>
  <div> CREDIT SUISSE BULLETIN 2 | 99</div>
  <pb number="24"/>
<!--...-->

<!--CS Bulletin 1999/2/fr-->

<!--...-->
<article n="12">
  <div heading="article"> «LES HOMMES</div>
</article>
<article n="13">
  <div heading="article"> OIN DE FRONTIÈRES»</div>
  <div> CHRISTIAN PFISTER</div>
  <div> Les frontières sont-elles un obstacle à la paix , ou</div>
  <div> CORNELIO SOMMARUGA , PRÉSIDENT DU COMITÉ INTERNATIONAL DE LA CROIX-ROUGE</div>
  <div> INTERVIEW : CHRISTIAN PFISTER , RÉDACTION BULLETIN</div>
  <div> au contraire les garantes de la paix ?</div>
  <div> CORNELIO SOMMARUGA</div>
  <div> Les deux sont vrais . L ' Europe , par</div>
  <div> exemple , s ' est développée parce que de nouveaux Etats n ' ont cessé d ' apparaî
  <div> 23</div>
  <div> CREDIT SUISSE BULLETIN 2 | 99</div>
  <pb number="24"/>
<!--...-->
```

Abbildung 16: Parallele de/fr Ausgaben des CS Bulletins 1999/2: Aligniert werden die Artikel 12(de)-13(fr)

3.2 Anweisungen

Für die Artikelalignierung müssen folgende Dateien im gleichen Verzeichnis stehen:

- Align.py: Python Script für die Alignierung
- settings.conf: Konfigurationsdatei für Anpassung des Input Pfads (Verzeichnis mit den parallelen XML-Dateien) und des Output Pfads (XML-Datei mit der Alignierungen)

- SettingsManager.py: Python Script für die Anbindung der Konfigurationsdatei

Bedingungen für die Ausführung von Align.py sind:

- Python 2.7⁸
- lxml⁹.

4 Satzalignierung

Vor der Satzalignierung mussten die XML-Dateien in das neue id Format konvertiert werden. Für diese Aufgabe habe ich das Perl Skript Convert_old_to_new_id_format.pl von Matin Volk benutzt. Damit alle XML Files Dateien automatisch konvertiert werden, habe ich das Shell Script convert_all_with_perl_script.sh geschrieben (vgl. Anhang 2).

Für die Satzalignierung wurden die Skripte aus dem Text&Berg Projekt (Autoren: Johannes Graën, Manuela Weibel und Fabienne Leuenberger) für sechs Sprachpaare (de-fr, de-en, de-it, fr-en, it-en, it-fr) angewendet. Damit die Satzalignierung leichter erfolgt, habe ich den Output der Artikelalignierung den Satzalignierungsskripten angepasst. Gleichzeitig hat Fabienne Leuenberger die Satzalignierungsskripte so erweitert, dass sie die Artikelalignierungen im TEI Format verarbeiten konnten. Außerdem hat Leuenberger, Anweisungen geschrieben, die mir bei der Ausführung der Satzalignierung geholfen haben (vgl. Anhang 2).

Bei den Artikeln, die nicht eins-zu-eins aligniert waren, hat die Satzalignierung nur die erste alignierte Artikelkombination genommen. Wir haben entschieden, weitere Anpassungen der Satzalignierungsskripten nicht auszuführen, weil die Skripte eigentlich für die Dateien gemeint sind, die die alignierten Artikeln in eins-zu-eins Form haben, wie es bei den richtig erkannten parallelen Artikel der Fall sein sollte. Die Ausgaben der Satzalignierung sind im Anhang 2 gespeichert.

5 Fazit

In diesem Bericht habe ich beschrieben, wie die Artikelerkennung und die Evaluation der PDF2XML Pipeline erfolgte und wie die anschliessende Artikel- und Satzalignie-

⁸ Python 2.7 Release in: <<https://www.python.org/download/releases/2.7/>>.

⁹ lxml in: <<http://lxml.de/>>.

rung durchgeführt worden sind. Nach der Analyse der Ergebnisse lässt sich folgendes schliessen:

- Wenn die PDF2XML Pipeline für die Artikelerkennung weiter verwenden werden sollte, müsste zuerst sichergestellt werden, dass die Schriftgrösse sowie die Fontnamen in allen parallelen Ausgaben der Zeitschrift konsistent sind;
- Wenn die Artikelerkennung für die Zwecke des Projekts Bulletin4Corpus von grösserer Bedeutung ist, sollte die Artikelerkennung bestenfalls mit einem anderen Modell durchgeführt werden, das anstatt der Schiftgrössen und Fontnamen andere Ankerpunkten verwendet. Dies könnten zum Beispiel die PDF-Seitennummer, an den die Artikel beginnen, sein.
- Wenn sich die Anzahl der alignierten Sätze für die Zwecke des Projekts als wichtiger als die Artikelerkennung erweist, sollten am besten nur die Satz-alignierungsskripte auf die Artikelalignierung mit dem Prinzip mehr-zu-eins angepasst werden, so dass auch parallele Sätze aus den Artikelalignierungen mehr-zu-eins gefunden werden.

6 Literaturverzeichnis

Salinger, Till (2015): Documentation of the PDF2XML Pipeline. Independent studies project, University of Zürich.