Fusion von Annotation und Präprozessierung als Vorschlag zur Behebung des Rohtextproblems

Jürgen Hermes, Christoph Benden

Die Nutzung sprachlicher Ressourcen aus dem WWW oder diversen Archiv-Datenträgern als Korpus für NLP-Anwendungen unterliegt technischen und rechtlichen Beschränkungen (Rohtextproblem). Das von Cunningham & Scott (2004) umrissene SALE-Framework ist Grundlage eines Lösungsvorschlags für die technischen Aspekte des Rohtextproblems in Form eines konsequent konfigurierbaren Präprozessors (SPre). Thematisiert werden unterschiedliche Aspekte der Konfiguration von Prozessketten und deren Komponenten. Die Einbindung von SPre in andere Anwendungen wird anhand zweier konkreter Implementierungen dargestellt.

The usage of linguistic Data available on the web as well as various archives and repositories as corpora for NLP applications is subject to technical and legal restrictions (raw text problem). The SALE-Framework outlined by Cunningham & Scott (2004) is taken as the basis for a solution to the technical aspects of the raw text problem for implementing a consistently configurable preprocessor (SPre). Various aspects of the configuration of process chains and their components are thematized. The integration of SPre in other applications is demonstrated by means of two specific implementations.

1. Rohe Texte

Unter einem Rohtext verstehen wir im folgenden einen digitalisiert vorliegenden, zugänglichen Text. "Zugänglich" bedeutet nicht ausschließlich, dass ein Zugriff über das WWW[1] erfolgen muss; es schließt hier natürlich auch direkten Zugriff auf Rohtexte ein (CD-Rom u.ä.). "Roh" in diesem Zusammenhang bezieht sich ausschließlich auf formale Merkmale des Textes und impliziert keinerlei inhaltliche Qualifikation. Ein perfektes rohtextverarbeitendes Programm würde dem Anwender einen Zugriff auf Rohtexte in ähnlicher Qualität wie bei der Verwendung manuell erstellter Korpora erlauben. Die Probleme im Zusammenhang mit der Verwendung von

[1] Im Rahmen linguistischer und NLP-Untersuchungen wird zunehmend das WWW als Korpus entdeckt und genutzt, vgl. Computational Linguistics 29/3 (September 2003 – Special Issue on the web as corpus).

Rohtexten als linguistischer Datenbasis werden in den folgenden drei Abschnitten diskutiert.

```
[...]□□□□□□□□□□□□□□$□□□□.□□□□□6□□□□□F□□□□□□□□□□□N□□□□□_□&□□□LAEN
kurz□IDEN T890531.119□QUEL TAZ□NR   2820□SEIT 20□DATU 31.05.1989□ZEIL
10□ART  Agentur□BEM  Hörfunk-Hinweis * in taz-Berlin: S.18□AUEB
Rundfunknachrichten in Friesisch}□BROT Bredstedt/Kiel (dpa) - Im
Rahmen der Sendereihe {Z_Fett Friesisch für alle}□     sendet der
Norddeutsche Rundfunk (NDR) über die "Welle Nord" im 1. □
Hörfunkprogramm jetzt jeden Dienstag um 19.15 Uhr Nachrichten in
friesischer□     Sprache. Die Meldungen werden vom NDR gemeinsam mit
dem Nordfriesischen□     Institut in Bredstedt (Kreis Nordfriesland)
erstellt. □□□□□□□□□□□□□□□□□'□□□□□□□□□□□□□□□□□□□□□□□□□□□□$□□□□.□□□□□6□□
□□□F□□□□□□□□□□□N□□□□□[□+□□□LAEN kurz□IDEN T890531.115□QUEL TAZ□NR
2820□SEIT 15□DATU 31.05.1989□ZEIL 12□ART  TAZ-Bericht□AUEB
Berichtigung-Betr.: Post aus der Moderne}□BROT Schlangen, Schwalben,
Hühnchen, ein Hurenkind (auf der ersten Seite, nicht□     auf der
zweiten!) [...]
```

Abb. 1: Rohtext, hier ausgelesen aus Archiv-CD Rom der taz 1986-1994

1.1 Technische Anforderungen

Die Prozessierung eines Textes wie dem in unserem Beispiel (Abb. 1) besteht zumindest aus drei Schritten: Konvertierung, Segmentierung und Annotation.

Ausgehend von einem solchen idiosynkratischen Format muss der eigentliche Text (Content) zunächst von seiner Formatierungsinformation (Form) getrennt werden. Dazu gehört auch die Rückübersetzung von Escape-Sequenzen beispielsweise für Umlaute. Es bietet sich an, den Ursprungstext in ein allgemein definiertes Format, etwa XML (Abb. 2), zu überführen, welches sich leicht mit bereits existierenden Tools oder Programmbibliotheken weiter prozessieren lässt.

```
<text date="31.05.1989" type="newspaper" source="taz" domain="misc"
  language="de">
<paragraph>Rundfunknachrichten in Friesisch</paragraph>
<paragraph>Bredstedt/Kiel (dpa) - Im Rahmen der Sendereihe Friesisch
  für alle sendet der Norddeutsche Rundfunk (NDR) über die
&amp;quot;Welle
  Nord&amp;quot; im 1. Hörfunkprogramm jetzt jeden Dienstag um 19.15 Uhr
  Nachrichten in friesischer Sprache. Die Meldungen werden vom NDR
  gemeinsam mit dem Nordfriesischen Institut in Bredstedt (Kreis
  Nordfriesland) erstellt.</paragraph>
</text>
```

Abb. 2 Konvertierter Rohtext im XML-Format

Anschließend kann die Segmentierung in linguistisch motivierte Einheiten erfolgen, worunter meist Tokenisierung und Satzgrenzenfestlegung verstanden werden. Auch in Schriftsprachen, die ihre Segmente durch Leerzeichen

(Wortgrenzen) oder definierte Zeichen (Satzenden) kennzeichnen, existieren ambige Konstruktionen, die Anforderungen hinsichtlich ihrer Behandlung an den Segmentierungsalgorithmus stellen (vgl. Mikheev 2003, Halama 2004). Ein Ausschnitt unseres Beispieltextes könnte nach dem Segmentierungsschritt wie in Abb. 3 dargestellt aussehen.

```
<?xml version="1.0" encoding="UTF-8"?>
<Text>
  <Sententoid>
    <Word><Characters>Rundfunknachrichten</Characters></Word>
    <Word><Characters>in</Characters></Word>
    <Word><Characters>Friesisch</Characters></Word>
  </Sententoid>
  <Sententoid>
    <Word><Characters>Bredstedt</Characters></Word>
    <Slash><Characters>/</Characters></Slash>
    <Word><Characters>Kiel</Characters></Word>
    <Bracket><Characters>(</Characters></Bracket>
    <Word><Characters>dpa</Characters></Word>
    <Bracket><Characters>)</Characters></Bracket>
  </Sententoid>
  <Sentence>
    <Word><Characters>Im</Characters></Word>
    <Word><Characters>Rahmen</Characters></Word>
    <Word><Characters>der</Characters></Word>
    <Word><Characters>Sendereihe</Characters></Word>
    <Word><Characters>Friesisch</Characters></Word>
    <Word><Characters>für</Characters></Word>
    <Word><Characters>alle</Characters></Word>
[...]
    <FullStop><Characters>.</Characters></FullStop>
  </Sentence>
[...]
</Text>
```

Abb. 3 Ausschnitt des segmentierten Textes

Für bestimmte Anwendungsszenarien kann zur Rohtextaufbereitung auch die Annotierung der durch die Segmentierung gewonnenen Einheiten zur Verfügung gestellt werden, etwa das Hinzufügen von Wortartklassifikationen (part of speech/POS tags), Satzmoduserkennung oder ähnliches. Streng genommen handelt es sich schon bei der üblichen Klassifizierung der Segmente (Wort, Satz etc.) um Annotation. Abb. 4 zeigt den Ausschnitt unseres Beispieltextes mit POS-Auszeichnungen.

```
<?xml version="1.0" encoding="UTF-8"?>
<Text>
 <Sententoid>
  <Word pos="SUB"><Characters>Rundfunknachrichten</Characters></Word>
  <Word pos="PRP"><Characters>in</Characters></Word>
  <Word pos="SUB"><Characters>Friesisch</Characters></Word>
 </Sententoid>
 <Sententoid>
  <Word pos="EIG"><Characters>Bredstedt</Characters></Word>
  <Slash><Characters>/</Characters></Slash>
  <Word pos="EIG"><Characters>Kiel</Characters></Word>
  <Bracket><Characters>(</Characters></Bracket>
  <Word pos="SUB"><Characters>dpa</Characters></Word>
  <Bracket> <Characters>)</Characters> </Bracket>
 <Sententoid>
 <Sentence>
  <Word pos="PRP"><Characters>Im</Characters></Word>
  <Word pos="SUB"><Characters>Rahmen</Characters></Word>
  <Word pos="ART"><Characters>der</Characters></Word>
  <Word pos="SUB"><Characters>Sendereihe</Characters></Word>
  <Word pos="SUB"><Characters>Friesisch</Characters></Word>
  <Word pos="PRP"><Characters>für</Characters></Word>
  <Word pos="IND"><Characters>alle</Characters></Word>
[...]
  <FullStop><Characters>.</Characters></FullStop>
 </Sentence>
[...]
</Text>
```

Abb. 4 POS-Annotierter Ausschnitt des segmentierten Textes

Bisweilen ist es nicht möglich, die beiden zuletzt angeführten Schritte sauber auseinanderzuhalten, da es sich sowohl bei der Segmentierung als auch bei der Annotation um einen inkrementellen Prozess handelt, bei dem kleinere Einheiten eventuell schon annotiert sein müssen, um die Grenzen größerer Einheiten zu ermitteln (man könnte z.B. für seinen Satzgrenzenerkennungs-algorithmus auf POS-Informationen zurückgreifen wollen).

1.2 Repräsentativität und Ausgeglichenheit

Ein Korpus ist repräsentativ bzw. ausgeglichen ("balanced"), wenn es

1. alle wesentlichen Momente der Sprachverwendung in korrekter Proportionalität erfasst und
2. erlaubt, die durch Einsatz des Korpus erzielten Ergebnisse zu generalisieren.

Forderung 1 ist mutmaßlich (vgl. McEnery 1996: 29ff, 77ff; Manning & Schütze 1999:118ff) oder sicherlich (Kilgarriff & Grefenstette 2003, 340ff) unerreichbar, zur Erfüllung der eigentlichen Forderung 2 aber unerlässlich. Während gewöhnlich auf eine möglichst feingranulare Textsortenwahl und

deren proportionale Gewichtung hingewiesen wird, führen Kilgarriff & Grefenstette (2003, 342) die gewaltige Quantität als ausgleichenden Faktor an: "The web is a dirty corpus, but expected usage is much more frequent than what might be considered noise". Die Generalisierbarkeit über die schiere Menge von Texten ist mit gegenwärtigen Korpusgrößen nicht zu gewährleisten; sie setzt den Einsatz von rohtextverarbeitenden Programmen und damit die Lösung des Rohtextproblems voraus.

1.3 Recht und Konsistenz: Juristische Voraussetzungen der Rohtextverwendung[2]

Rohtexte sind nicht frei hinsichtlich ihres Zugriffs und ihrer Verarbeitung, sie unterliegen in Deutschland automatisch dem Urheberrechtsgesetz (UrhG, Stand 10.9.2003) ohne die Notwendigkeit einer entsprechenden Angabe durch den Autor. Auch wenn das Gesetz selbst den juristischen Laien (hier: uns) schwer zugänglich ist, zeigen die z.B. vom Aktionsbündnis "Urheberrecht für Bildung und Wissenschaft"[3] aufgeworfenen Fragen und gesammelten Forderungen, was dem Gesetz nach z.Zt. nicht oder nur in einer rechtlichen Grauzone durchführbar ist. So ist es nicht rechtssicher möglich, dass "öffentlich geförderte wissenschaftliche Einrichtungen [...] digitale Dokumente für den internen Gebrauch elektronisch archivieren dürfen", wobei "digitale Dokumente" sich hier auf Rohtexte jeder Art bezieht. Vollkommen undiskutiert ist die Frage, inwieweit eine ‘uneigentliche’, indirekte Verwertung von Texten überhaupt einen Sachverhalt des UrhG darstellt, z.B. die NLP-orientierte Auswertung v.a. grammatischer und formaler Eigenschaften von Texten.

Möglicherweise lässt sich aus dem für wissenschaftlichen Belange immer angeführten §52a UrhG ("Öffentliche Zugänglichmachung für Unterricht und Forschung") hinsichtlich Bereitstellung und § 53 UrhG ("Vervielfältigungen zum privaten und sonstigen eigenen Gebrauch") hinsichtlich Archivierung eine Berechtigung zur dauerhaften Speicherung und Bereitstellung von Rohtexten ableiten. Allerdings enthält § 53 zwei zusätzliche Pferdefüße: Erstens dürfen die kopierten Rohtexte "weder unmittelbar noch mittelbar Erwerbszwecken dienen", was für zu entwickelnde Technologien, die idealiter mittelfristig industrielle Anwendung finden, schwer zu gewährleisten ist. Zweitens dürfen

---

[2] Man beachte, dass sich die folgenden Ausführungen auf Texte deutscher Urheber bzw. solche Texte beziehen, die eine "Inländerbehandlung" nach der "Berner Übereinkunft" zulassen (paraphrasiert: der ausländische Text wird so behandelt, als unterläge er dem deutschen Urheberrecht).

[3] http://www.urheberrechtsbuendnis.de.

die Texte nur vervielfältigt werden, "soweit nicht zur Vervielfältigung eine offensichtlich rechtswidrig hergestellte Vorlage verwendet wird", was bei einer semi- bzw. vollautomatischen Rohtexterhebung, v.a. im Kontext des WWW, unüberprüfbar ist.[4]

Ein Fazit scheint uns unter den gegebenen Umständen nicht möglich. Eine umfassende Infrastruktur (Verwertungsgesellschaften) zur Vergütung von Inhalten des WWW existiert bislang nicht, so dass hier auch guter Wille nicht weiterhilft. Für erworbenes Material (z.B. in Form von CD-Rom-Archiven) scheint die wichtigste Frage zu sein, ob rechtmäßiger Gebrauch derselben ausschließlich über die – für NLP-Zwecke gewöhnlich unzureichenden – mitgelieferten Werkzeuge erfolgen darf oder ob rohtextverarbeitende Programme zum Einsatz kommen dürfen (was eine weitere Archivierung im Idealfall überflüssig macht).

Mit der rechtlichen Problematik rund um Vervielfältigung einher geht das Problem der Konsistenz eines rohtextbasierten Korpus. Stammen Teile des Korpus aus dem WWW, so kann aufgrund permanenter Fluktuation von Inhalt und Zugänglichkeit der Webseiten nicht sichergestellt werden, dass Untersuchungen, Verweise und Rohtext noch dem Stand einer bestimmten Erhebung entsprechen. Für wissenschaftliche Zwecke ist eine lokale Persistierung von Rohtexten aus dem WWW unerlässlich, da der Vergleich und die Feinabstimmung unterschiedlicher Verfahren nur durch Anwendung auf ein und dieselbe Datenbasis möglich sind.

2. SALE: Konfigurierbare Komponenten in Prozessketten

Seit den 1990er Jahren gibt es Bestrebungen, Softwarewerkzeuge für linguistische Anwendungen zu erstellen, die nicht ausschließlich auf ein Anwendungsszenario hin, sondern in Hinsicht auf Nachvollziehbarkeit und Wiederverwendbarkeit angelegt werden. In *Natural Language Engineering* (10 3/4) werden eine Reihe von Systemen bzw. Werkzeugen vorgestellt, die unter

[4] Es sei noch erwähnt, dass gerade die genannten Paragraphen, die eine Archivierung zu Forschungszwecken zumindest partiell erlauben, im Rahmen der geplanten Novellierung des UrhG aktiv diskutiert werden. V.a. hinsichtlich digitaler Vervielfältigung/ Weiterverarbeitung sind eine Reihe Neuerungen zu erwarten, vgl. den Referentenentwurf des Bundesministerium für Justiz (http://www.bmj.bund.de/media/archive/760.pdf), synoptisch präsentiert in Jani (2004). Ferner verliert lt. § 137k der § 52a, wichtige Grundlage eines Sonderstatus für Forschung und Lehre hinsichtlich des Urheberrechts, ab dem 31.12.2006 regulär seine Wirkung.

diesem Paradigma, genannt "Software Architecture for Language Engineering" (SALE, Cunningham/Scott 2004, im einleitenden Artikel zu dieser Ausgabe), entwickelt wurden. So verschieden die einzelnen Ansätze auch sein mögen, gemeinsam haben sie, dass sie zum einen sprachliche Ressourcen[5] (oder Sprachsignale[6]) konsumieren und produzieren und zum anderen im weitesten Sinne Prozesskettensysteme sind oder sich in solche implementieren lassen.

Prozessketten bestehen aus einzelnen Komponenten, die über definierte Schnittstellen miteinander kommunizieren und den Untersuchungsgegenstand, das Sprachsignal, inkrementell mit Annotation anreichern. Annotationen, die von Komponenten in der Prozesskette produziert werden, können von den darauffolgenden Komponenten konsumiert werden, um eine weitergehende Analyse durchzuführen. Im Vergleich zu monolithischen Systemen ergeben sich Vorteile hinsichtlich der Robustheit, der Skalierbarkeit und der Konfigurierbarkeit (vgl. Neff, Byrd & Boguraev 2004, 307), bestehende Komponenten können wiederverwendet und mit neu entwickelten in einer Prozesskette kombiniert werden.

Die Kombinierbarkeit von Komponenten erfordert eine definierte Schnittstelle, als die sich das Sprachsignal anbietet, da jede Komponente auf dieses Annotationen konsumierend und/oder produzierend zugreift. Bemühungen hinsichtlich eines internationalen Standards für ein Framework zur linguistischen Annotation werden gegenwärtig von der ISO/TC 37/SC 4 Arbeitsgruppe 1, basierend auf XML, RDF(S) und OWL, vorangetrieben[7]: "[The] goal is to develop a framework for the design and implementation of linguistic resource formats and processes in order to facilitate the exchange of information between language processing modules" (Ide & Romary 2004: 216). Gängige Systeme, z.B. GATE[8], nutzen sogenannte Annotationsgraphen, wie sie z.B. im ATLAS-Format (Bird, Day, Garafolo, Henderson, Laprum & Liberman 2000) definiert werden. In einem solchen Graph sind Sprachsignal und

---

[5] Vgl. Cunningham et al. (2005): "Language Resource (LR): refers to data-only resources such as lexicons, corpora, thesauri or ontologies. Some LRs come with software (e.g. Wordnet has both a user query interface and C and Prolog APIs), but where this is only a means of accessing the underlying data we will still define such resources as LRs."

[6] Der Begriff Signal entspricht den im ATLAS-Format formulierten Konzept, s.u.

[7] http://www.tc37cs4.org/

[8] Das System GATE (General Architecture for Text Engineering, http://www.gate.ac.uk) wurde in Java implementiert und in Hinsicht auf Generalisierbarkeit optimiert. Als Beispiel für ein industriell motiviertes System, in dem die Performance im Vordergrund steht, lässt sich TEXTTRACT (Neff, M., Byrd, R., Boguraev 2004), ein von IBM betreutes und in C++ umgesetztes Prozesskettensystem anführen.

Annotation – sei es nun Formatinformation oder linguistische Auszeichnung – getrennt ("stand-of markup"). Im Signal befinden sich sogenannte Anker, zwischen denen sich Annotationen (als Kanten des Graphen) erstrecken.

Im Rahmen der Rohtextverarbeitung mit Prozessketten kommt der externen Konfiguration der Komponenten zentrale Bedeutung zu. Zum einen dienen unterschiedliche Konfigurationsdateien als Ersatz der Persistierung von (komplexer) Tokenisierung und Annotierung von Rohtexten. Zum anderen – und dieser Punkt ist nicht zu überschätzen – stellen Konfigurationsdateien in Bezug zu den von ihnen gesteuerten Komponenten (und deren bekannter Funktionalität) eine kommunizierbare Dokumentation textwissenschaftlicher Arbeit dar. Hierzu sollten die Konfigurationsdateien idealiter folgenden Bedingungen genügen[9]:

- Die Konfiguration ist **konsequent**, d.h. alle relevanten Schritte der Komponente sind manipulierbar. Dies ist eine reziproke Forderung, da sie ein entsprechend umfassend konfigurierbare Komponente erfordert.
- Die Konfiguration ist "**dokumentativ**" bzw. selbsterklärend, indem sprechendes Vokabular für Klassifikationen bzw. Operationen verwendet und der Prozessablauf in der Anordnung reflektiert wird.
- Die Konfiguration ist **abstrakt**, d.h. sie sollte ohne Programmierkenntnisse versteh- und manipulierbar sein, um die Konfiguration unabhängig von der Implementierung zu halten.

Aus der obigen Diskussion ergibt sich ein generelles Modell einer SALE-Komponenete, die neben ihrem operativen Teil – den Algorithmen – den Zugriff auf ein genormtes Datenformat – hier realisiert als Annotation Graph – und einen Interpreter für externe Konfiguration zur Verfügung stellen muss (Abb. 5)

[9] Rollenorientierte Funktionsbeschreibungen für Komponenten könnten den Abstraktionsgrad sogar vollständig von der Implementierung trennen (Lutz Schneidermeier, p.c.); dies würde aber eine einheitliche Architektur aller Komponenten zumindest hinsichtlich der Konfiguration erfordern, mithin engere Kollaboration der u.U. weltweit verteilten Entwickler.

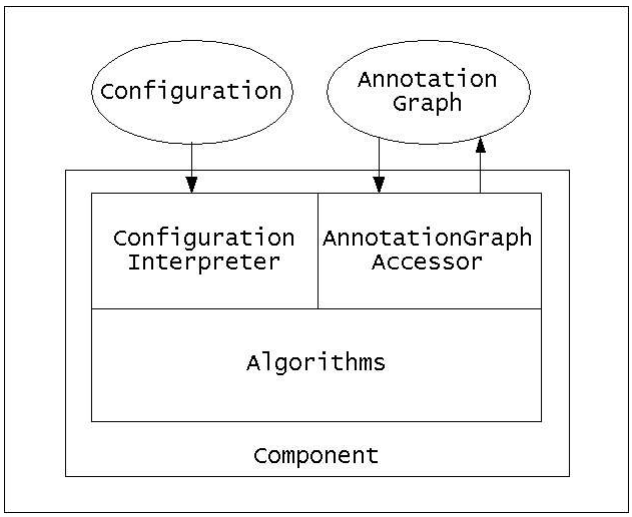

Abb. 5 SALE-Komponentenmodell

3. Rohtextverarbeitung im SALE-Paradigma: SPre

Der Rohtextprozessor SPre[10] (in Entwicklung seit Februar 2004) setzt die oben diskutierten Komponentenmerkmale um (vgl. Abb. 6). Sowohl das Gesamtsystem, wie auch die einzelnen Komponenten (Reader, Parser, Annotator, Persistor) realisieren das in Abb. 5 spezifizierte Modell. Zentraler Bestandteil ist ein Plugin-Manager, der gemäß einer externen Konfiguration Rohtexte durch eine Prozesskette präprozessiert und annotiert.

[10] Für weitere Details zur Architektur siehe auch Benden & Hermes (2004)

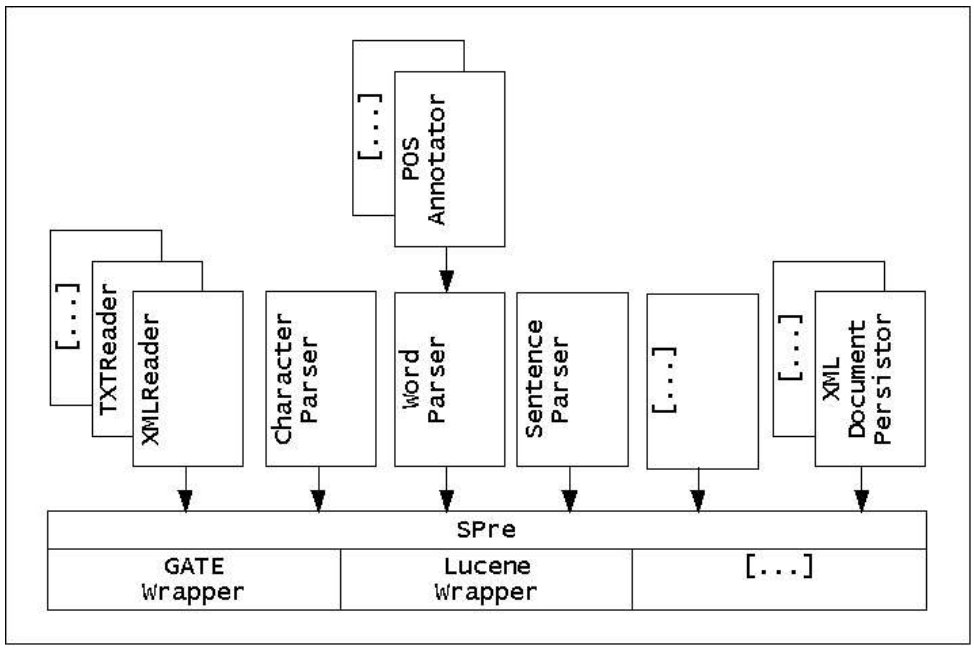

Abb. 6 Architektur des Präprozessors SPre

Die externe Konfiguration wird in XML formuliert und besteht aus einem Manifest, das die gewünschte Kette von Parsern in der Sequenz ihrer Anwendung definiert (Abb. 7), und einem operativen Teil (partiell in Abb. 8), der Klassifikationen und Operationen auf den einzelnen Layern definiert.

```
<Manifest>
  <InputFileTypes>
    <FileType> XML </FileType>
    <FileType> TXT </FileType>
  </InputFileTypes>
  <ParserSequence>
    <Parser> <Name> CharacterParser </Name> </Parser>
    <Parser> <Name> WordParser </Name>
             <Input> CharacterLayer </Input></Parser>
    <Parser><Name> SentenceParser </Name>
            <Input> WordLayer </Input></Parser>
  </ParserSequence>
  <Persistors>
    <Persistor> XMLDocumentPersistor </Persistor>
  </Persistors>
</Manifest>
```

Abb. 7 Manifest der externen SPre-Konfiguration

Operative Definitionen erfolgen über ein Vokabular von Klassifikatoren (`Item`, `StartsWith`, `Contains`, `EndsWith`, `ContainsOnly`) und dem Operator `Merge` zur Auflösung von Ambiguitäten. Das folgende Fragment löst im Kontext `Word` den auf der Ebene des `WordParser` ambigen Punkt dergestalt auf, dass (1) ein Punkt nach links zum (letzten) Buchstaben (eines Wortes) gezogen wird, unter der Bedingung, dass die sich ergebende Sequenz

als Abkürzung gelistet ist; dass (2) ein Punkt innerhalb eines Wortes verbleibt, wenn kein `WhiteSpace` nach dem Punkt folgt.[11]

```
<Ambiguity>
 <Element>Dot</Element>
 <Merge Type="left">                                                    (1)
  <Sequence>
    <Element>Letter</Element>
    <Element>Dot</Element>
    <Element>WhiteSpace</Element>
  </Sequence>
  <Conditions><Condition>isAbbreviation</Condition></Conditions>
 </Merge>
 <Merge Type="leftright">                                               (2)
  <Sequence>
    <Element>Letter</Element>
    <Element>Dot</Element>
    <Element>Letter</Element>
   </Sequence>
   <Conditions/>
 </Merge>
</Ambiguity>
```

Abb. 8 Ausschnitt aus dem operationalen Teil der SPre-Konfiguration

Konkrete Kenntnis der Implementierung des Prozessierungsvokabulars sind unnötig, um die verwendeten Algorithmen zur Disambiguierung von Punkten abzulesen oder auszudrücken. Entsprechende Definitionen benötigen Punkte in Numeralausdrücken, Verwendung von Bindestrichen am Zeilenende (als Trennungs- bzw. Gedankenstrich) etc. Das gesamte klassifizierende und algorithmisch umgesetzte Wissen von SPre wird in der externen Konfiguration leicht lesbar gebündelt. Der Anspruch an Robustheit bzw. Fehlertoleranz[12] wird u.E. am besten dreistufig realisiert:

1. Unbekannte Items werden als `Unprocessable` markiert und bleiben ansonsten unbeachtet.[13] Unprocessables werden auf allen höheren Ebenen von SPre ignoriert.
2. Unprocessables werden mit Kontext und Quell-URL in einer Logdatei festgehalten. Eine beliebige Anzahl von Texten kann verarbeitet werden, wobei die Fehler protokolliert werden.

---

[11] Die allgemeine Definition des Characters `Dot` als `FullStop` auf Wortebene klassifiziert den orthographischen Punkt ansonsten als Satzendzeichen.

[12] Robustheit wird hier nur aus der Perspektive des Verhältnisses von aktueller Konfiguration und Rohtext betrachtet. Fehlerhafte Implementierung, defekte Rohtextdateien u.ä. müssen natürlich gleichfalls behandelt werden, liegen jedoch unterhalb der betrachteten Ebene.

[13] Die Qualität der Prozessierung ist nicht notwendig schlechter, wenn Unprocessables auftreten, da sie oft Material enthalten, das marginal ist (orthographische Anführungen, undefiniert Character aus der OCR etc.) und nicht in die Analyse eingehen sollte.

3. Werden die Fehler protokolliert und SPre mit einem geeigneten Interface ausgeführt, können die Unprocessables in die Konfigurationsdatei integriert werden. Hierzu werden die in SPre implementierten Klassifikatoren und Operatoren verwendet.

4. Fallstudien

Um verschiedene Möglichkeiten für die Nutzung von SPre zu erproben, wurden zwei Implementierungen verfolgt: Zum einen wurde SPre als Processing Resource in das GATE-System integriert, um es als Komponente in einer Processing Pipeline zu testen. Zum anderen wurde die Interaktion mit einer unabhängigen Applikation anhand der Suchmaschine Lucene erprobt, der SPre als `Analyzer`-Komponente zuarbeitet.

4.1 SPre als Processing Resource im GATE-System

Das Framework GATE ist ein bekanntes, wohldokumentiertes[14], lauffähiges und ständig weiterentwickeltes[15] System, dessen Annotationsformat weitgehend kompatibel mit dem oben vorgestellten ATLAS-Format ist. Um selbsterstellte Komponenten im GATE-System zu nutzen, muss man diese als Processing Resource in das GATE-Komponentenmodell einbinden. Dazu ist es nötig, die Komponente als Java Bean zu wrappen, um die Kommunikation zwischen dem steuernden System und der einzubindenden Komponente zu ermöglichen. In der Wrapperklasse wird sichergestellt, dass eine Instanz von SPre angelegt und gestartet wird und dieser die benötigten Ressourcen zur Verfügung gestellt werden (Setzen von Pfaden zur XML-Konfigurationsdatei, zu den zu prozessierenden Dateien etc.). Weiterhin werden die von SPre prozessierten Annotationen in den Annotationsgraphen, der vom GATE-System für den prozessierten Text angelegt wurde, übertragen und können damit innerhalb der graphischen Oberfläche angezeigt werden. Eine Version von SPre als GATE-Komponente ist inzwischen als Download verfügbar.[16]

---

[14] System und Tutorials finden sich auf der Seite www.gate.ac.uk.

[15] Die aktuellen Entwicklungen zielen vor allem darauf ab, neue Herausforderungen, die durch die Forschungen am Semantic Web, großen digitalen Bibliotheken und Sprachanalyse durch maschinelles Lernen aufgeworfen werden, gerecht zu werden.

[16] System und einführendes Tutorial unter http://www.spinfo.uni-koeln.de/forschung/spre

4.2 SPre als Analyzer für die Suchmaschine Apache Lucene

Die Open-Source Suchmaschine Apache Lucene[17] verwendet für Indexierung und Abfrage einen vorgeschalteten Präprozesser (Analyzer), der wahlweise einfach alle durch `WhiteSpace` getrennte Tokens liefert oder in unterschiedlich hohem Grad Stopwortanalyse und Stammformreduktion durchführt. Der integrierte `GermanAnalyzer` führt eine Tokenisierung mit nachgeschalteter algorithmischen Stammformreduktion[18] durch, deren Qualität aus linguistischer Sicht unbefriedigend ist[19].

Die Verwendung von SPre als Analyzer für Lucene kann als Ausgangspunkt einer NLP-Verwendung der Suchmaschine betrachtet werden. Die bislang nicht verwendeten `type`-Felder[20] der einzelnen Tokens von Lucene nehmen die Klassifizierungen der Parser und Annotoren von SPre auf und dienen wahlweise als Filter bei Suchanfragen. In der Verwendung als `TokenFilter` ist SPre wesentlich flexibler als die vorhandenen `TokenFilter`, da neue, unerwartete oder fremdsprachige Character leicht über eine Änderung der Konfiguration integrierbar sind. Anders als die vorhandenen Analyzer kann SPre beliebige Tokens, z.B. auch n-Gramme, Sätze etc. zur Indexierung geben. Das Ziel ist eine Integration von linguistisch orientierter Präprozessierung in eine Suchmaschine, um damit die Nutzung von Rohtexten zu erleichtern.

---

[17] http://jakarta.apache.org/lucene, vgl. auch Hardt & Theis 2004

[18] "Algorithmisch" ist hier im Gegensatz zu "lexikonbasiert" zu verstehen.

[19] Vgl. auch Hardt & Theis 2004, 58ff. Je nach Anwendung ist eine linguistisch korrekte Stammformreduktion eher wichtig (NLP mit nachfolgender Verarbeitung, Linguistik) oder eher marginal (Text Retrieval, Kategorisierung etc., vgl. Jackson & Moulinier 2002:11ff).

[20] Jedes Token von Lucene speichert einen typbezeichnenden String, der bisher ausschließlich den Defaultwert `word` annimmt und ungenutzt bleibt.

5. Fazit

Erhärten sich die Annahmen von Kilgarrif & Grefenstette (2004), dass angestrebte Repräsentativität von Korpora durch Verwendung extrem vergrößerter Textmengen ersetzt werden kann, so wird ein Kompetenzaufbau in der maschinellen (Prä)prozessierung von Rohtexten eine der zentralen Herausforderungen für die Computerlinguistik in den nächsten Jahren sein. Rohtextverarbeitung erfordert Programme, die kleinschrittige Verarbeitung ermöglichen, was im Zusammenhang mit aktuellen Überlegungen zur Architektur von NLP-Komponenten eine allgemeine Bauform sprachverarbeitender Programme in Form von konfigurierbaren Prozessketten nahelegt. SPre stellt unseren Diskussionsbeitrag zu diesem Problemfeld dar.

Literaturverzeichnis

Benden, C. & Hermes, J. (2004). Präprozessierung mit Nebenwirkungen: Dynamische Annotation. In *Beiträge zur 7. Konferenz zur Verarbeitung natürlicher Sprache (KONVENS)*, Wien, 25-28.

Bird, S, Day, D., Garofolo, J., Henderson, J., Laprun, C. & Liberman, M. (2000). ATLAS: A Flexible and Extensible Architecture for Linguistic Annotation. In *Proceedings of the Second International Conference on Language Resources and Evaluation*, 1699-1706.

Bontcheva, B., Tablan, V., Maynhard, D. & Cunningham, H. (2004). Evolving GATE to Meet New challenges in Language Engineering. *Natural Language Engineering 10(3/4)*, 211-225.

Cunningham, H. et al. (2005, 6. Januar): Developing Language Processing Components with GATE (a User Guide). For GATE version 3 beta 1 (July 2004). Zugegriffen am 15.01.2005 über http://www.gate.ac.uk/sale/tao/index.html

Cunningham, H. & Scott, D. (2004). Software Architecture for Language Engineering. *Natural Language Engineering 10(3/4)*, 205-209.

Halama, A. (2004). Flache Satzverarbeitung. In K.-U. Carstensen, C. Ebert, C. Endriss, S. Jekat, R. Klabunde, & H. Langer (eds.), *Computerlinguistik und Sprachtechnologie: Eine Einführung* (2. überarbeitete und erweiterte Auflage, 218-231). Heidelberg: Spektrum.

Hardt, M. & Theis, F. (2004). *Suchmaschinen entwickeln mit Apache Lucene.* Frankfurt: Software und Support Verlag.

Ide, N. & Romary, L. (2004). International standard for a linguistic annotation framework. *Natural Language Engineering 10(3/4)*, 211-225.

Jackson, P. & Moulinier, I. (2002). *Natural Language Processing for Online Applications. Text Retrieval, Extraction and Categorization*. Amsterdam: Benjamins.

Jani, O. (2004, 29. September). Gegenüberstellung von UrhG und Referentenentwurf Zugegriffen am 7.1.2005 über *http://www.urheberrecht.org/topic/Korb-2/syn/Korb_2_RefE_Synopse_short_01.pdf*

Kilgarrif, A. & Grefenstette, G. (2003). Introduction to the Special Issue on the Web as Corpus. *Computational Linguistics 29(3)*, 333-347.

Manning, C. & Schütze, H. (1999). *Foundations of Statistical Natural Language Processing.* Cambridge, MA: MIT Press.

McEnery, T. & Wilson, A. (1996). *Corpus Linguistics. An Introduction.* 2. Auflage. Edingburgh: University Press.

Mikheev, A. (2003). Text Segmentation. In R. Mitkov (ed.), *The Oxford Handbook of Computational Linguistics* (201-218). Oxford: University Press.

Neff, M., Byrd, R. & Boguraev, B. (2004). The Talent system: TEXTRACT architecture and data model. *Natural Language Engineering 10(3/4)*, 307-325.