Text Processing

Version 2006-07-30/1
Arbeitspaket AP 1
verantwortlicher Partner: FH Worms

TextGrid

Modulare Plattform für verteilte und kooperative wissenschaftliche Textdatenverarbeitung - ein Community-Grid für die Geisteswissenschaften

Bundesministerium für Bildung und Forschung

Projekt: **TextGrid**
Teil des D-Grid Verbundes und der deutschen e-Science Initiative
BMBF Förderkennzeichen: 07TG01A-H
Laufzeit: Februar 2006 - Januar 2009

Dokumentstatus: final
Verfügbarkeit: öffentlich
Autoren:
Christoph Ludwig, FH Worms
Andrea Zielinski, IDS Mannheim

1. Einleitung

Im Rahmen des Arbeitspaketes 1 wurden vom TextGrid-Projekt im 1. Halbjahr 2006 zahlreiche Literaturbeiträge und existierende Werkzeuge für die wissenschaftliche Textverarbeitung gesichtet und auf ihre Relevanz für TextGrid hin evaluiert. Dieser Bericht schildert die wichtigsten Ergebnisse für die im Projektantrag genannten Module Tokenizer, Kollationiere, Lemmatisierer, Sortierer, XML-Editor und OCR sowie einige für TextGrid generell wichtige Ergebnisse der Literaturrecherche.

2. Ergebnisse der Literaturrecherche

In der Literatur fanden sich sehr viele philologische Arbeiten, die mit Software-Unterstützung erzielte Ergebnisse präsentieren, die Konsequenzen und Möglichkeiten für die Textwissenschaften reflektieren, welche sich aus der Verbreitung des Mediums und Werkzeuges Computer ergeben, und die Webportale zu elektronisch erschlossenen Textkorpora vorstellen. Aber nur ein auffallend kleiner Anteil der Literatur befasst sich explizit mit den Anforderungen an die benutzten Werkzeuge, die aus den Fragestellungen der Textwissenschaftler folgen.

Dennoch konnten aus den wenigen gefunden Arbeiten, die konkrete Anforderungen nennen, sowie aus diversen Beschreibungen durchgeführter oder geplanter Editionsprojekte für TextGrid wertvolle Informationen gewonnen werden. Beispielsweise kommt Rischer [Ris2003] bei seinen Überlegungen, wie angesichts der extrem kurzen Produktzyklen im IT-Bereich die langfristige Nutzbarkeit von Editionen gesichert werden kann, zu Ergebnissen, die TextGrid in seinem Ansatz bestärken: Nur durch die Verwendung offener Standards, portabler Software usw. sieht Rischer die notwendige Autarkie gesichert. Dies wird von TextGrid umgesetzt, indem es mit Globus Toolkit auf eine offene Grid-Plattform baut, den selbst entwickelten Code unter den Open Source Lizenzen GPL bzw. LGPL veröffentlicht sowie die Daten allgemein im XML-Format und speziell Annotationen im verbreiteten, ebenfalls offenen XML/TEI-Format speichert.

Im ARCHway-Projekt [KJDP2005] entstand Software, die zwar dem Arbeiten mit lokal verfügbaren Daten verhaftet blieb und nicht die Möglichkeiten zum Zugriff auf verteilte Textkorpora in einem Grid ausnutzt, aber wie TextGrid eine modulare Plattform für verschiedenartigste textwissenschaftliche Werkzeuge bildet. Dieses Projekt hat demonstriert, dass das Eclipse SDK nicht nur für die Softwareentwicklung, sondern auch als ergonomische Umgebung für textwissenschaftliche Werkzeuge geeignet ist und aufgrund seines Plug-in Konzeptes für modulare Anwendungen prädestiniert ist. Es steht deshalb zu erwarten, dass sich Eclispe auch für das GUI zu den TextGrid-Modulen bewähren wird.

Bei der Literaturrecherche fanden sich wie erwartet Anwendungen, die Werkzeuge erforden, die zu implementieren den Rahmen des laufenden TextGrid-Projektes sprengen würde. TextGrid hat den Anspruch, dass Fachwissenschaftler die fehlenden textwissenschaftlichen oder linguistischen Werkzeuge (wie sie z.B. in [LL2005], [EBBR2005] oder [SGK+2000] beschrieben werden) ohne großen Aufwand in die von TextGrid realisierte Workbench integrieren können. Aber es ist wünschenswert, dass sich TextGrid auch als Plattform für computergestütztes Arbeiten in den Nachbardisziplinen der Sprachwissenschaften etabliert und deshalb Anwendungen wie das Verlinken von Notentext mit den korrespondierendn Abschnitten in einer Einspielung im Falle einer musikwissenschaftlichen Edition potentiell unterstützt. Hiervon könnten beispielsweise eine Edition des Arnold Schönberg Archivs [Mux2006] oder eine angedachte Edition des Werks von Bob Dylan

[Joh2003] profitieren. TextGrid muss deshalb darauf achten, dass solche Module, die nicht ausschließlich auf Text und eingescannten Abbildungen der Quellen operieren, von der Architektur des entwickelten Systems nicht ausgeschlossen werden. Tatsächlich ließen sich Module zur Integration von Audio- und Videodaten auch für textwissenschaftliche Arbeiten nutzen: Beispielsweie stehen mit Aufzeichnungen von Theateraufführungen, die ein zeitgenössischer Autor selbst inszeniert hat, oder Lesungen zusätzliche autorisierte Quellen zur Verfügung, die nur elektronisch adäquat wiedergegeben werden können.

3. Anforderungsanalyse

Wie schon erwähnt gibt es nur wenige Literaturbeiträge, die explizit Bedingungen an die Werkzeuge auflisten. Die im Folgenden genannten Anforderungen sind deshalb überwiegend Folgerungen aus den in der Literatur beschriebenen Anwendungen bzw. Endprodukten.

3.1 XML-Editoren

Der XML-Editor ist voraussichtlich das Werkzeug, über das die Fachwissenschaftler – rein zeitlich gesehen – am meisten mit TextGrid interagieren. Es ist deshalb sehr wichtig, dass der Editor effizientes und produktives Arbeiten unterstützt, auch dann, wenn der Nutzer kein ausgewiesener XML-Experte ist.

Abgesehen von den allgemein gültigen Empfehlungen für GUI-Designs bedeutet dies:

- Der Editor darf ausschließlich valide XML-Dokumente abspeichern. Idealerweise warnt er den Anwender schon bei der Eingabe, wenn diese ein invalides XML-Dokument nach sich zieht.
- Der Editor sollte es ermöglichen, für die momentane Aufgabe irrelevante XML-Elemente bzw. Attribute auszublenden. Es sollte unkompliziert zu konfigurieren sein, welche Aufgaben die Sichtbarkeit welcher Elemente und Attribute benötigen.
- Aus Gründen der Erweiterbarkeit und Flexibilität muss der Editor für verschiedenste XML-Formate konfiguriert werden können. Er sollte deshalb DTD, RelaxNG sowie XML Schema unterstützen. Unabhängig von der Unterstützung für andere Formate muss der Editor den Anwender unbedingt bei der Eingabe von XML-kodiertem TEI-Markup zur Seite stehen, z.B. durch kontextsensitive Hilfe, „intelligente“ Vorschläge zur Vervollständigung von Eingaben etc.
- Für mache Aufgaben ist es hilfreich, einen Bezug zwischen der aktuellen Position im XML-Dokument und einem Faksimile-Abbild der Quelle herstellen zu können – beispielsweise, wenn Schadstellen oder Streichungen im Manuskript notiert werden. Für solche Anwendungen sollte der Editor in dem XML-Dokument enthaltene Links auswerten und den jeweils passenden Ausschnitt des Faksimiles einblenden können. (Vgl. hierzu auch [KJDP2005].) Dies erfordert möglicherweise eine gewisse Integration mit dem in diesem Bericht noch nicht betrachteten Link-Editor.
- XML erzwingt, dass alle Elemente eines Dokumentes einen Baum bilden, d.h. ein Element A ist entweder vollständig in einem Element B enthalten oder A befindet sich völlig außerhalb von B; teilweise überlappende XML-Elemente gibt es nicht. Bei der Annotation von

Transkripten treten Überlappungen hingegen regelmäßig auf: Häufig begegnen einem Streichungen, die sich über einen Zeilenumbruch hinweg erstrecken. Wenn also sowohl die Zeilen als auch die Streichungen als eigene XML-Elemente repräsentiert werden sollen, ergibt sich ein Konflikt.
Hierfür sind verschiedene Lösungen vorgeschlagen worden, auch im Rahmen von TEI [SMB2004, Chapter 31] . All diesen Lösungen ist gemein, dass sie die Komplexität des resultierenden XML-Dokumentes erheblich erhöhen, so dass dessen Integrität bei manuellen Modifikationen nur schwer aufrecht zu halten ist. Idealerweise erlaubt der XML-Editor, die „störenden“ Auszeichnungen zu ignorieren und auf einem vergleichsweise simplen XML-Dokument zu arbeiten. Erst beim Abspeichern fügt der Editor die ignorierten Auszeichnungen wieder hinzu und führt zugleich alle Transformationen aus, die notwendig sind, um nach wie vor valides XML zu erhalten.

- Eine letzte Anforderung ergibt sich nicht zwingend aus der Literatur, sondern basiert auf einer in TextGrid getroffenen Architektur-Entscheidung: Der XML-Editor soll als Plug-in in das Eclipse SDK integrierbar sein.

3.2 Lemmatisierung

Das Lemmatisierungstool stellt unterschiedliche Funktionalitäten bereit, die im Rahmen von TextGrid benötigt werden. Einerseits ist es ein Baustein der Textanalyse, welcher zur Aufbereitung historischer Korpora verwendet werden kann, und andererseits dient es dazu, die Suchfunktionalität auf Korpora und/oder Wörterbüchern zu verbessern.

Benötigte Module für die Textanalyse:

- Morphologische Analyse und Lemmatisierung von historischen Korpora. Liefert als Ausgabe einen morphologisch annotierten Korpus;
- Aufbau des Wörterbuches:
  - Integration vorhandener Wörterbücher
  - Automatische Generierung von neuen Lexikoneinträgen (korpusgestützt)[1]

Benötigte Module für die Suche:

- Lemmatisierung der Suchanfrage; Wörterbuch-Funktion (Dictionary Look-up)
- Erkennung von Varianten und Homonymen

Der Benutzer kann entweder a) den gesamten Text morphologisch analysieren oder b) ein einzelnes Wort im Text markieren, für das automatisch der entsprechende Lexikoneintrag angezeigt wird, auch wenn das Wort in der flektierten Form im Text auftritt. Hierzu kann eine beliebige Anzahl von Wörterbüchern eingebunden werden.

Basisanforderungen an das Lemmatisierungstool sind

[1] Zusätzlich soll ein benutzerfreundliches Terminologieverwaltungstool zur manuellen Erstellung von Lexikoneinträgen verwendet werden

- hohe Performanz durch Aufbau bzw. Einbindung großer Lexika
- Wortbildungsanalyse für unbekannte Wörter
- Erkennung von Varianten (sprachliche Varianten, Rechtschreibvarianten, stilistische und graphemische Varianten)
- Einbindung in die Suche
- Kompatibilität mit dem Tokenizer
- Verarbeitung unterschiedlicher Textformate (Unicode, UTF-8, etc.)

Folgende Funktionalitäten werden nicht bereitgestellt:

- Tool zur Verwaltung von Lexikoneinträgen
- kontextsensitive Analyse (POS-Tagging)
- Erkennung von lexikalisch-semantischen Varianten
- Erkennung von komplexen Eigennamen (Bsp: Firma B&B, Italia)
- Testsuite für die Evaluierung des Lemmatisierungstools

Da Programme zur Lemmatisierung nicht nur sehr rechen- und speicherintensiv sind, sondern auch einen relativ hohen Aufwand zum Aufbau eines Basislexikon erfordern, ist eine Bereitstellung dieser Semantic Grid Applikation äußerst erstrebenswert.

3.3 Kollationierung

Eine elektronische Edition ermöglicht es, grundsätzlich alle in den benutzten Quellen festgestellten Varianten darzustellen; der Platz ist nicht länger notwendig ein limitierender Faktor. Aber neben der reinen Dokumentation fällt einer Edition auch die Aufgabe zu, die Textgenese für den Leser (oder besser: Nutzer) nachvollziehbar zu machen. Wenn sämtliche Varianten gleichgewichtig dargestellt werden, so sinkt die Übersichtlichkeit rapide mit der Zahl der Quellen. Auch bei einer elektronischen Edition muss sich eine Editor also im Allgemeinen entscheiden, welche Variationen er als signifikant betrachtet. Der Editor muss deshalb einem Kollationierer in TextGrid Regeln übergeben können, welche Unterschiede als nachrangig behandelt werden sollen. (Etwa wenn der Editor der Schreibung eines Wortes mit dem Graphem *u* oder dem Graphem *v* keine besondere Bedeutung beimisst.) Als Teil dieser Regeln muss der Kollationierer auch Ausnahmelisten unterstützen.

Für einen Kollationierer ist es sehr aufwändig, völlig autonom zu erkennen, dass in einzelnen Quellen ganze Textblöcke ausgelassen oder gar verschoben wurden, speziell wenn die korrespondierenden Blöcke noch zusätzliche mehr oder weniger zahlreiche Varianten enthalten. Deshalb soll der für TextGrid zu entwickelnde Kollationierer sog. Aufsetzpunkte verarbeiten können, also Markierungen, von denen ausgehend der Vergleich aller Fassungen wieder synchronisiert wird. Weil diese Aufsetzpunkte nicht für alle Fassungen in der gleichen Reihenfolge defininiert sein müssen, kann der Editor damit dem Kollationierer Verschiebungen größerer Textblöcke manuell mitteilen.

Ähnlich wie bei der Lemmatisierung sind in der Literatur verschiedene Vergleichs-Algorithmen beschrieben. In einer der neuesten Arbeiten schlagen Spencer und Howe [SH2004] eine Methode vor, die ohne Festlegung auf einen Basistext auskommt und auf Verfahren aus der Bioinformatik aufbaut. Auch hier lässt sich nicht aus dem Ergebnis der Literaturrecherche ableiten, dass TextGrid einem der

vorgeschlagenen Algorithmen den Vorzug geben sollte, weshalb die Entscheidung, welche Kollationierungsmethode als erste implementiert wird, nach pragmatischen Gesichtspunkten getroffen werden kann.

3.4 Tokenizer

Der Tokenizer soll die verschiedenen Texteinheiten in einem Werk identifizieren und markieren. Von jedem Tokenizer wird erwartet, dass er Wort- und Satzgrenzen erkennt; wünschenswert ist auch eine automatische Auszeichnung von größeren Einheiten wie Absätzen und Kapiteln. Für letztere muss es möglich sein, dem Tokenizer Regeln mitzugeben, anhand derer Kapitelgrenzen erkennbar sind, beispielsweise an einer immer in Fettdruck, mit wenigstens zwei Zeilen Abstand nach oben gesetzten Überschrift.

Abgesehen von Ausnahmelisten, in denen einzelne Fehlerkennungen korrigiert werden können, muss ein in TextGrid sinnvoll einsetzbarer Tokenizer auch benutzerdefinierte Sonderregeln unterstützen, mit denen Festlegungen für andernfalls mehrdeutige Interpunktionen im Falle von Abkürzungen o.ä. getroffen werden.

In der Literatur werden auch Anwendungen bzw. Werkzuge beschrieben, die weitergehende Sprach- bzw. Texteinheiten unterscheiden: Beaudouin und Won [BW1996] zerlegen die gefundenen Wörter noch weiter in ihre Silben, um die Metrik des vorliegenden Textes – in ihrem Fall alexandrinische Verse – systematisch untersuchen zu können. Wolfrum [Wol2003] nutzt graphische Elemente der Handschrift – in dem untersuchten Beispiel gezeichnete Rahmen, die durch Tintenfraß zur Fragmentierung des Manuskripts führten – um Gedichte zu Einheiten zusammenzufassen. Wir können uns vorstellen, dass beide Ansätze auch in anderen Projekten eine sinnvolle Anwendung finden können; aber sie scheinen uns nicht hinreichend allgemein zu sein, um notwendig von einem ersten Tokenizer für TextGrid unterstützt zu werden. Es handelt sich eher um typische Beispiele für potentielle von TextGrid-Anwendern entwickelte Erweiterungsmodule.

3.5 Sortierer

Wenn ein Textdokument – beispielsweise ein Index oder eine Sammlung von Wörterbuchartikeln – sortiert werden soll, so muss das entsprechende Tool zunächst die Möglichkeit bieten, Sortierfelder und -schlüssel zu definieren, Stoppwörter anzugeben, die bei der Sortierung ignoriert werden, und nicht zuletzt die gewünschte Sortierreihenfolge festzulegen.

Die in vielen Editionsprojekten bewährten Module #SORTIERE bzw. #SVORBEREITE aus TUSTEP verlangen, dass das Sortieralphabet für jeden Schlüssel explizit angegeben wird. Dies ist angesicht der Fülle der in Unicode zur Verfügung stehenden Zeichen nicht mehr zeitgemäß; von einem modernen Sortierer kann man erwarten, dass die Auswahl des Sortieralphabetes durch die Angabe eines Kulturraumes möglich ist; genauer gesagt durch die Benennung der Lokale, welche die Sortiergepflogenheiten des entsprechenden Kulturraumes wie in den einschlägigen Standards festgelegt beschreibt [Küs2001]. Weil es immer noch Sonderfälle oder in den Standards nicht erfasste Sonderzeichen geben kann, muss es ferner möglich sein, die durch die Lokale gegebenen Sortierregeln zu ergänzen oder modifizieren.

3.6 OCR

Ziel eines in TextGrid entwickelten OCR-Moduls kann es nicht sein, die eigentlichen Erkennungsalgorithmen zu implementieren, die ein gescanntes Bild in ein Textdokument wandeln. Vielmehr geht es um die darauf notwendigerweise folgende Qualitätsanreicherung und -kontrolle.

Hierfür werden grundsätzlich zwei Ansätze verfolgt: Einerseits kann versucht werden, den erkannten Text wieder in der gleichen Schrifttype, den gleichen Einzügen etc. wie das Original zu setzen und das Ergebnis mit dem Scan der Originalvorlage zu vergleichen. Dieses Verfahren erwartet dort (und nur dort) größere Abweichungen zwischen den Bildern, wo im OCR-Schritt ein Zeichen falsch erkannt wurde. Wenn die benötigten Fonts zur Verfügung stehen und die Generierung der Differenzbilder sehr sorgfältig optimiert wird, kann diese Methode einen Korrektor sehr gezielt zu den Stellen im Dokument führen, die einer manuellen Kontrolle bedürfen. Die Implementierung eines solchen Moduls für TextGrid wäre also zu erwägen, wenn mit den konventionellen Verfahren nicht die gewünschte Minimierung der Fehlerquote erzielt werden kann.

Im Gegensatz zu dem soeben beschriebenen, auf Bildmanipulation und -auswertung beruhenden Ansatz versuchen linguistische Verfahren, unplausible Schreibungen zu erkennen und Alternativvorschläge zu generieren. Hierfür werden typischerweise die erkannten Texte mit umfangreichen Wörterbüchern abgeglichen; ausgefeilte Scoring-Mechanismen berücksichtigen die Häufigkeit bestimmter Wörter u.ä. bei der Plausibilitätsentscheidung. Ein solches Tool soll in TextGrid zur Verfügung gestellt werden, wobei der Anwender die berücksichtigten Wörterbücher, Namensverzeichnisse etc. vorgeben können soll.

Dieser Ansatz versagt, wenn der vorgelegte Text zu viele Wörter enthält, die nicht in den benutzten Wörterbüchern enthalten sind. Strohmeier et al. [SRSM2003] berichten von guten Ergebnissen bei dem Versuch, Wörterbücher mit Wortlisten zu ergänzen, die aus einer Sammlung fachspezifischer Webseiten generiert wurden. Allerdings nutzen die von den Projektpartnern in TextGrid eingebrachten Textkorpora (z.B. Romane von Jean Paul) nicht die moderne Sprache, die sich auf den zwangsläufig zeitgenössischen Webseiten findet. Der Nutzen dieses Ansatzes wäre deshalb im Rahmen von TextGrid sehr beschränkt, weshalb auf eine Umsetzung vorerst verzichtet werden kann.

Die selben Autoren haben auch eine Methode vorgestellt [SRSM2003a], wie die auf Wörterbüchern basierende Fehlersuche durch eine zusätzliche interaktive Lernphase deutlich verbessert werden kann. Die Integration einer solchen Lernphase in TextGrids Modul zur OCR-Korrektur ist zu erwägen.

4. Evaluierte Software

In AP 1 sichteten die TextGrid-Mitglieder bereits weit über 100 Programme und Bibliotheken, um sie im Hinblick auf auf ihre Nutzbarkeit in TextGrid oder auf vorbildhafte Features zu evaluieren.

4.1 XML-Editoren

Auf dem Markt befinden sich zahllose XML-Editoren von teils sehr unterschiedlicher Qualität. Eine umfassende Bestandsaufnahme wäre an dieser Stelle ob des Aufwandes nicht zu leisten. Wir mussten uns deshalb darauf beschränken, die bekanntesten Editoren bzw. solche, auf die wir durch Verweise seitens ähnlicher Projekte aufmerksam wurden, zu betrachten.

Unterstützung für die Validation gegen DTD, RelaxNG sowie XML Schema, eine integrierte XSLT engine, Suche in den Dokumenten mittels XPath etc. findet sich heute bei den meisten „großen“ XML-Editoren. Programme wie XML Spy, <oXygen/> oder Exchanger XML Lite sind allerdings nicht unter einer Open Source Lizenz verfügbar und scheiden deshalb für eine Nutzung in TextGrid aus. aXe und Butterfly können nicht die für TextGrid geforderte Integration in Eclipse bieten.

Somit bleiben VEX, Jaxe und EPT. Die Stärke von VEX ist, dass es die Dokumente über Style Sheets (CSS) gestaltet präsentieren kann, statt den häufig unübersichtlichen Wust von XML-Elementen. Ähnliches kann auch mit Jaxe erreicht werden (XHTML via XSLT oder über vom benutzer angepasste Grafikausgabe), das insgesamt über mehr Features zu verfügen und leichter konfigurierbar scheint. EPT, das aus dem ARCHway-Projekt hervorging, ist schließlich ein Sonderfall, weil es selbst bereits eine Zusammenstellung von Plug-ins für Eclipse ist. Es hat sich bereits im textwissenschaftlichen Kontext bewährt und hat eine sehr modulare Struktur, so dass es wahrscheinlich erscheint, dass zumindest Teile von EPT für TextGrid nutzbar sind, evtl. nach einigen Anpassungen.

4.2 Lemmatisierung

Insbesondere für das Deutsche stehen derzeit keine geeigneten Lemmatisierungstools zur Verfügung, deren Quellcode im Rahmen eines Open-Source-Projektes weiterentwickelt werden könnte.

Da die morphologische Analyse auch Ergebnisse für linguistische Varianten – insbesondere historische und dialektale Wortformen – liefern soll, ist eine Extension des Basisalgorithmus und des Lexikons jedoch unumgänglich.

State-of-the-Art Lemmatisierer für flexionsreiche Sprachen basieren in der Regel auf der Two-Level-Morphology [Kos1984]. Sowohl die Lexika, in welchem Informationen über Stämme und Affixe kodiert sind, sowie die Regeln für die Zerlegung komplexer Wörter, lassen sich dabei mittels Werkzeugen aus dem Bereich der Finite State Transducer, den gewichteten Transduktoren, modellieren.

Lemmatisierungssysteme, die dagegen auf Vollformenlexika basieren, bilden keine produktiven Wortbildungsregeln ab, und scheitern bei der Analyse neu gebildeter Wortformen.

Leider sind derzeit keine adäquaten Lexika mit entsprechenden Flexionsinformationen, die die Grundlage einer erfolgreichen Lemmatisierung bilden, frei verfügbar. Als möglicher Ausweg bietet sich daher eine Lernkomponente an, die umfangreiche Korpora syntaktisch analysiert und daraufhin neue Lexikoneinträge erzeugen kann. Ein derartiges Tool, z.B. der Durm Lemmatizer (Praharshana Perera and René Witt), ist kürzlich als open-source-Projekt von der Uni Karlsruhe entwickelt worden. Hierbei konnte für die Korpusanalyse das open-Source-Tool aus dem GATE Projekt (Sheffield) sowie der TreeTagger (Helmut Schmid) der Uni Stuttgart eingesetzt werden.

Die Evaluierung kommerziell verfügbarer Tools ergab ein relativ einheitliches Bild: Folgende Funktionalitäten sind standardmäßig enthalten: a) Flexionsanalyse mit Ausgabe der morphologischen Information, b) heuristische Wortbildungs- und Derivationsanalyse mit Ausgabe möglicher Wortbildungsmusters, c) umfangreiche Lexika (ca. 100.00 bis 250.000 Lexem-Einträge), d) Generierungskomponente, e) Ergänzen von benutzerdefinierten Lexikoneinträgen.

Die kommerziellen Tools XeLDA® ,WMTrans und GERTWOL (mit Preisen in der Größenordnung von ca. 10.000 Euro) erzielen die besten Analyseergebnisse dank eines hochwertigen internen Lexikons und elaborierter Wortbildungsheuristiken. Auf einer Stichprobe von 5.000 Wörtern

(ausgenommen ca. 500 Fremdwörter und Rechtschreibfehler) werden nahezu alle bekannten Wörter korrekt lemmatisiert (Erkennungsrate: 98%). Die Auswertung bezogen auf die Analyse unbekannter Wörter (mithilfe der generativen Wortbildungskomponente) gelingt dagegen nur in ca. 70 % der Fälle. Häufig werden mehrere Wortanalysen generiert, von denen nicht immer die erste korrekt ist. Dies könnte zwar prinzipiell durch die benutzergesteuerte Erweiterung des Lexikons verbessert werden, doch ist dies aufgrund einer unzureichenden Dokumentation der morphosyntaktischen Klassen und mangelnder Unterstützung bei der Terminologieverarbeitung nur mühsam zu bewerkstelligen.

Weitere Lemmatisierungstools, die im Rahmen von Drittmittelprojekten entstanden sind, und teilweise unter GNU General Public License verwendet werden können, sind Morphix (DFKI, G.Neumann), DeKo (IMS, Stuttgart), Deutsche Malaga-Morphologie (DMM) (Uni Erlangen-Nürnberg), Morphy (W.Lezius), Project Deutscher Wortschatz (Uni Leipzig), TAGH (Berlin-Brandenburgische Akademie der Wissenschaft) und das CISLEX (Uni München).

Mit dem Lemmatisierungstool aus dem Projekt 'Tares, das von der Uni Trier in TextGrid eingebracht wird, steht bereits ein Werkzeug zur Verfügung, um historische Texte, bei denen noch kein Thesaurus vorliegt, interaktiv bzw. halbautomatisch zu lemmatisieren. Dabei wird nach und nach ein Thesaurus aufgebaut, der dann für weitere Lemmatisierungen genutzt werden kann.

4.3 Kollationierung

Juxta ist ein Java-Programm zur Kollationierung. Es bietet ein sehr übersichtliches GUI zur Darstellung der gefundenen Variationen. Das Programm wurde erst in diesem Jahr Open Source, könnte aber seitdem wohl grundsätzlich dahingehend angepasst werden, als Plug-in in Eclipse integriert zu werden. Der benutzte Algorithmus ist eine Portierung des in der Unix-Welt altbekannten „Arbeitspferdes“ *diff*, aber es liegen erst wenig Erfahrungen vor, wie gute Ergebnisse Juxta im Vergleich zu anderen Programmen erzielt.

Collate ist sehr populär und verfügt über Features wie Ausgabe von TEI-konformem XML, die es eigentlich für TextGrid sehr interessant machen würden. Leider ist Collate bislang nur für die Macintosh-Plattform unter einer kommerziellen Lizenz verfügbar, so dass TextGrid es nicht einsetzen kann.

Das bereits bei den XML-Editoren erwähnte EPT bietet auch Module für die Kollationierung, die sich vermutlich für TextGrid nutzen lassen.

4.4 Tokenizer

TAPoRware ist eine Sammlung von Tools zur Textanalyse von der McMaster University, die u.a. auch Tokenizer enthält. Allerdings scheinen die Tools ausschließlich die Eingabe über Webmasken zu unterstützen, was für TextGrid angesichts der großen bearbeiteten Korpora keine brauchbare Option ist. Links zum Download der Software führen ins Leere, so dass auch keine Aussage über die Implementierung gemacht werden kann.

CEM (Colloquial Entropy Markup) ist ein unter GPL veröffentlichtes Java-Paket für die Auszeichnung von Texten anhand von Markov-Modellen. Es wurde beispielsweise für Aufgaben wie Bibliographie-Extraktion, Namensextraktion, Segmentierung von Chinesischem Text etc. benutzt. CEM kann nicht einen herkömmlichen Tokenizer für TextGrid ersetzen, aber es könnte einen solchen evtl. sinnvoll ergänzen.

4.5 Sortierer

Die Suche nach Sortierprogrammen ergab – abgesehen von den für unsere Zwecke untauglichen Unix Systemtools sort, uniq etc. – lediglich die in TUSTEP enthaltenen Module #SORTIERE bzw. #SVORBEREITE und #SAUFBEREITE. TUSTEP unterliegt einer kommerziellen Lizenz und kann deshalb in TextGrid nicht genutzt werden. Es wird deshalb notwendig sein, für TextGrid ein eigenes Sortiermodul zu entwickeln.

4.6 OCR

PrimeOCR ist ein kommerzielles Programme für Optical Character Recognition. Gamera ist ein Programm für den gleichen Zweck, das unter der GNU General Public License vertrieben wird. Choudhury et al. [CLF+2006] berichten, dass PrimeOCR bei Frakturschriften mitunter Probleme habe, während Gamera sehr gut trainierbar sei.

Weil für TextGrid keine Lizenzen kommerzieller Programme beschafft werden sollen, scheidet PrimeOCR für die Verwendung in TextGrid vorerst aus. Evtl. kann in TextGrid Gamera eingesetzt werden.

Für die OCR-Kontrolle wird ein eigenes Modul entwickelt werden, evtl. unter Einbeziehung der in [SRSM2003a] beschriebenen Software, sofern deren Autoren zu einer Veröffentlichung unter einer Open Source Lizenz bereit sind.

5. Zusammenfassung der Ergebnisse

Die Sichtung der einschlägigen Literatur zeigte die breite Spannweite an philologischen Anwendungen, die Forscher entwickeln und für die TextGrid idealerweise ein geeignetes Framework bietet soll. Auch wenn es nicht im Rahmen dieses Projektes möglich sein wird, für all diese Anwendungen die notwendigen Module in TextGrid bereitzustellen, so müssen sie doch jetzt schon bei der Planung von TextGrid bedacht werden, um nicht künftige Lösungen zu verbauen – allein dafür lohnte sich bereits der für die Sichtung notwendige Aufwand, weil diese Punkte für die spätere Akzeptanz in den textwissenschaftlichen Communities mit entscheidend sein werden.

Darüber hinaus ergaben sich dabei auch für TextGrid wertvolle Hinweise auf frühere Softwareprojekte (ARCHway, GATE), die sich zwar noch nicht die Möglichkeiten des Grid Computings zu Nutze machten, aber ebenfalls einen modularen Ansatz verfolgten und z.T. ähnliche Frameworks zur GUI-Gestaltung einsetzten, wie sie für TextGrid geplant sind. TextGrid plant, mit den Entwicklern dieser Projekte Kontakt aufzunehmen, um aus den dort gemachten Erfahrungen zu lernen. Ob es auch möglich sein wird, einzelne der dort entstandenen Module nach TextGrid zu portieren, lässt sich zum jetzigen Zeitpunkt noch nicht abschätzen.

Anhang A: Bibliographie

[Alt2006] Altova Ges.m.b.H.: *XML Spy 2006*. Homepage http://www.altova.com/products/xmlspy/xml_editor.html.

[ARP2006] Applied Research in Patacriticism: *Juxta*. Homepage: http://www.patacriticism.org/juxta/.

[BBAW2005] Berlin-Brandenburgische Akademie der Wissenschaften: *TAGH*. Homepage: http://www.dwds.de/erschliessung/morphologie.

[BGJ2006] G. Braungart, P. Gendolla, F. Jannidis (Hrsg.): *Jahrbuch der Computerphilologie 7 (2005)*. mentis Verlag, Paderborn, 2006.

[But2004] The Butterfly XML Team: *Butterfly XML*. Homepage http://butterflyxml.org/.

[BW1996] V. Beaudouin und F. Won: *The Metrometer: a Tool for Analysing French Verse.* Literary & Linguistic Computing, 1996, 11:23-31. Verfügbar auf http://tinyurl.com/ewn6a.

[Can2006] Canoo Engineering AG: *WMTrans*. Homepage http://www.canoo.com/wmtrans/products/index.html.

[CBT+2006] H. Cunningham, K. Bontcheva, V. Tablan, D. Maynard et al.: *GATE - General Architecture for Text Engineering*. Homepage http://gate.ac.uk/.

[Cla2006] Cladonia Ltd.: *Exchanger XML Lite Editor Version 3.2*. Homepage http://www.freexmleditor.com/.

[CLF+2006] G. S. Choudhury, T. DiLauro, R. Ferguson, M. Droettboom und I. Fujinaga: *Document Recognition for a Million Books.* D-Lib Magazine, 2006, 12(3): doi:10.1045/march2006-choudhury.

[DFF2006] M. Droettboom, R. Ferguson und I. Fujinaga: *Gamera 3.0.1*. Homepage: http://ldp.library.jhu.edu/projects/gamera.

[EBBR2005] J. H. Esling, L. D. Bettany, A. Benner und S. Rose: *Phonetic Structure and Aquisition of Laryngeal and Pharyngeal Articulations*. Text Technologies, 14(1): 21-31. Verfügbar auf http://texttechnology.mcmaster.ca/pdf/vol14_1_03.pdf.

[GTD+2006] D. Guillaume, B. Tasche, C. Dedieu, O. Kykal, L. Guillon, B. Delacretaz uns S. Kitschke: *Jaxe*. Homepage http://jaxe.sourceforge.net/.

[HHRS2003] C. Henkes, W. Hettche, G. Radecke, und E. Senne (Hrsg.): *Schrift - Text – Edition.* Band 19 der Beihefte zu editio, Niemeyer, Tübingen, 2003.

[IMS2001] Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart: *DeKo - Derivations- und Kompositionsmorphologie*. Homepage: http://www.ims.uni-stuttgart.de/projekte/DeKo/index_english.shtml.

[Joh2003] J. John: *Back to poetry, oder: Wie ediert man einen* performing artist*?* In: Henkes et al. [HHRS2003], 273-284.

[KAA+2006] J. Krasnay, M. Andersson, J. Austin, A. Berkovits, M. Gautier und D. Holroyd: *VEX - A Visual Editor for XML*. Homepage http://vex.sourceforge.net/.

[KJDP2005] K. Kiernan, J. W. Jaromczyk, A. Dekhtyar und D. C. Porter mit K. Hawley, S. Bodapati und I. E. Iacob: *The ARCHway Project: Architecture for Research in Computing for Humanities through Research, Teaching, and Learning.* Literary & Linguistic Computing, 2005; 20: 69-88. Verfügbar auf http://tinyurl.com/l5h82.

[Küs2001] M. W. Küster: *Die "European Ordering Rules" (EOR, ENV 13710), Einheitliche Regeln für das Sortieren multilingualer Daten.* In: Literary & Linguistic Computing, 2001; 16: 330 -338 . Auch verfügbar auf http://www.zdv.uni-tuebingen.de/tustep/prot/prot782-eor.html.

[KEP2004] Kompetenzzentrum für elektronische Erschließungs- und Publikationsverfahren in den Geisteswissenschaften an der Universität Trier: *TAReS*. Homepage: http://www.mhdwb.uni-trier.de/TAReS/index.html.

[Kos1984] K. Koskenniemi: *A General Computational Model for Word-form Recognition and Production*. In: Proceedings of the 22nd annual meeting on Association for Computational Linguistics, ACM, 1984: 178-181. Verfügbar auf http://portal.acm.org/citation.cfm?id=980529.

[Lez2001] W. Lezius: *Morphy*. Homepage: http://www.wolfganglezius.de/doku.php?id=public:cl:morphy.

[Lin1966] Lingsoft, Inc.: *GERTWOL: Morphologisches Analysesystem für das Deutsche*. Homepage: http://www2.lingsoft.fi/cgi-bin/gertwol/.

[LL2005] C. Labbé, D. Labbé: *A Tool for Literary Studies: Intertextual Distance and Tree Classification*. Literary & Linguistic Computing, 2005. Verfügbar auf http://tinyurl.com/mh6lj.

[LMO1996] S. Langer, P. Maier, J. Oesterle: *CISLEX*. Homepage: http://www.cis.uni-muenchen.de/projects/CISLEX/complexabs.html.

[Lor1996] O. Lorenz: *Die Deutsche Malaga-Morphologie (DMM)*. Homepage: http://www.linguistik.uni-erlangen.de/~orlorenz/DMM/DMM.html.

[Mux2006] T. Muxeneder: *Archivierung und virtuelle Edition – Arnold Schönbergs Nachlass als offenes Archiv*. In: Braungart et al. [BGJ2006], 53-66.

[Neu2001] G. Neumann: *Morphix - A Fast and Portable Morphological Component for Inflectional Languages*. Homepage: http://www.dfki.de/~neumann/morphix/morphix.html.

[PR2006] Prime Recognition, Inc.: *PrimeOCR Version 4.2*. Homepage: http://www.primerecognition.com/.

[PW2006] P. Perera und R. Witt: *Durm Lemmatizer*. Homepage http://www.ipd.uka.de/~durm/tm/lemma/.

[Ris2003] T. Rische: Eine säurefreie Edition des Ullysses. In: Henkes et al. [HHRS2003], 339-348.

[Rob1994] P. Robinson: *Collate - Interactive Collation of Large Textual Traditions.* Beschrieben auf http://sql.uleth.ca/dmorgwiki/index.php/COLLATE_Text_editing_software.

[Sch2006] H. Schmid: *TreeTagger Version 3.1*. Homepage http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html.

[SGK+2000] W. B. Seales, J. Griffioen, K. Kiernan, C. J. Yuan und L. Cantara: *The Digital Atheneum: New Technologies for Restoring and Preserving Old Documents*. Computers in Libraries, 2000, 20:2, 26-30. Verfügbar auf http://www.infotoday.com/cilmag/feb00/seales.htm.

[SH2004] M. Spencer und C. Howe: *Collating Texts Using Progressive Multiple Alignment*. Computers and the Humanities, 2004, 38 (3): 255-270. Verfügbar auf http://www.springerlink.com/openurl.asp?genre=article&id=doi:10.1007/s10579-004-8682-1.

[SMB2004] C. M. Sperberg-McQueen und L. Bernard (Eds.): *TEI P4 - Guidelines for Electronic Text Encoding and Interchange*. The TEI Consortium, 2004. Verfügbar auf http://www.tei-c.org/P4X/index.html.

[SRSM2003] C. Strohmaier, C. Ringlstetter, K. U. Schulz und S. Mihov: *Lexical Postcorrection of OCR-Results:The Web as a Dynamic Secondary Dictionary?*. In: Proceedings of the Seventh International Conference on Document Analysis and Recognition, IEEE, 2003, 1133-1137.

[SRSM2003a] C. Strohmaier, C. Ringlstetter, K. U. Schulz und S. Mihov: *A Visual and Interactive Tool for Optimizing Lexical Postcorrection of OCR-Results*. In: Proceedings of the Conference on Computer Vision and Pattern Recognition Workshop, IEEE, 2003, 32. Verfügbar auf http://www.cis.uni-muenchen.de/people/Schulz/Pub/DIAR03.pdf.

[Syn2006] SyncRO soft Ltd.: *<oXygen/> 7.2*. Homepage http://www.oxygenxml.com/.

[TAP2005] The TAPorWare Project: *TAPorWare 1.1*. Homepage: http://taporware.mcmaster.ca/.

[Tem2005] Temis Intelligence: *XeLDA*. Homepage http://www.temis-group.com/index.php?id=124&selt=1.

[Wol2003] U. Wolfrum: *Thomas Kybbets* The teares of time. In: Henkes et al. [HHRS2003], 331-338.

[WP2005] P. Wellens und W. Le Page: *aXe – Advanced XML Editor*. Homepage http://www.adrem.ua.ac.be/~wellenslepage/index.php.

[Yea2001] S. Yeates: *CEM - Colloquial Entropy Markup*. Homepage: http://www.advogato.org/proj/CEM/.

[ZDV2006] ZDV der Uni Tübingen (K. Schälkle, W. Ott, H. Fuchs): *Tustep Version 2006*. Homepage http://www.zdv.uni-tuebingen.de/tustep/.