Exposé zur Diplomarbeit

Entwicklung eines (Meta-)Modells für linguistisch annotierte Daten

Betreuer: Ulf Leser
Anke Lüdeling

Autor: Florian Zipser
eMail: zipser@informatik.hu-berlin.de

1 Motivation

Die Korpuslinguistik untersucht sprachliche Phänomene anhand empirischer Daten. Um möglichst genaue Regeln und Besonderheiten der zu untersuchenden Texte erkennen und benennen zu können, ist es notwendig, eine geeignete Stichprobe auszuwählen und zu analysieren. Diese Stichprobe wird, bezogen auf einen bestimmten sprachlichen Kontext, Korpus genannt. Ein Korpus bzw. die in ihm enthaltenen Texte werden auf vielfältige Weise annotiert, wobei sich die Art der Annotationen je nach dem Ziel der Forschung unterscheiden. Aufgrund der Größe typischer Korpora ist es meistens notwendig, ihre Analyse durch rechnergestützte Systeme zu unterstützen.
Eine Herausforderung ist die Darstellung und Persistenzierung annotierter Korpora. Hierzu hat sich eine Vielzahl von Annotationsmodellen und -formaten etabliert, die in der Regel aber nur für einen bestimmten Anwendungsfall entwickelt wurden und auch nur für diesen geeignet sind. Ein Annotationsmodell ist eine, nicht an eine Technik gebundene, Abstraktion der Daten. Ein Annotationsformat ist die Persistenzierung der Daten, sowie die Instanziierung eines Annotationsmodells.
Für den Korpuslinguisten stellt sich zu Beginn der Arbeit oft die Frage, auf welches Modell/ Format er sich stützen soll. Ist die Entscheidung einmal gefallen, ist er später oft gezwungen, inhaltliche Entscheidungen aufgrund der technischen Eigenschaften des gewählten Formats (und der damit kompatiblen Werkzeuge) zu treffen. Das kann sich stark einschränkend auf die Arbeit auswirken, da die Festlegung auf ein Format oft nicht ohne großen Aufwand geändert werden kann.
An dieser Stelle setzt die hier beschriebene Arbeit an. Sie untersucht die Möglichkeit der Transformation annotierter Korpora zwischen unterschiedlichen Modellen.

2 Zielsetzung

Ziel dieser Diplomarbeit ist es, verschiedene Annotationsmodelle zu untersuchen, die ihnen zugrunde liegenden Konzepte zu identifizieren und in einem gemeinsamen (Meta-)Modell abzubilden. Dieses (Meta-)Modell muss mächtig genug sein, um jedes einzelne Konzept „hinreichend genau“ repräsentieren zu können.
Anschließend soll auf dieser Grundlage ein Framework entwickelt werden, dass es ermöglicht, die untersuchten Annotationsmodelle aufeinander abzubilden (im Sinne eines „Universalkonverters“). Dabei soll kein direktes 1:1 Mapping von Modell A auf Modell B entstehen. Das entwickelte (Meta-)Modell fungiert als Schnittstelle, sodass ein Mapping von Modell A auf das (Meta-)Modell und von diesem auf Modell B entsteht. Dies hat den Vorteil, dass die Anzahl der unterschiedlichen Mapper deutlich geringer ist als bei einem 1:1 Ansatz.
Um die Möglichkeit offen zu halten, neben den untersuchten Modellen weitere Modelle in die konvertierbare Menge mit aufzunehmen, soll das Framework so gestaltet und umgesetzt werden, dass für das Hinzufügen weiterer Modelle klar definierte und gut dokumentierte Schnittstellen zur Verfügung stehen.
Ein weiteres Ziel dieser Arbeit ist es, den Prozess der Konvertierung möglichst performant zu gestalten. Da die Konvertierung nur einen Zwischenschritt in der linguistischen Arbeit darstellt

(bspw. zwischen der Erzeugung und der Visualisierung eines Korpus), soll der Zeitverbrauch angemessen sein. Hierfür soll die Möglichkeit der Parallelisierung einzelner Prozesse untersucht und gegebenenfalls umgesetzt werden.

3 Herangehensweise

3.1 Modelle/ Formate

Es muss eine geeignete Menge an möglichst unterschiedlichen Annotationsmodellen bestimmt und bzgl. der in ihr repräsentierbaren Konzepte analysiert werden. Oft jedoch liegen Annotationsformate ohne eine Modellbeschreibung vor. In solchen Fällen muss zunächst eine Abstraktion eines Formats zu einem Modell getroffen werden. Ein wichtiger Aspekt bei der Bestimmung der Modelle ist das Vorhandensein eines Pools an Daten im betreffenden Modell/ Format. Anhand der realen Daten kann später das (Meta-)Modell auf seine Mächtigkeit und Abdeckung der Konzepte geprüft werden. In dieser Arbeit betrachtete Modelle/ Formate sind u.a. PAULA [1], TIGER [2] und EXMARaLDA [3].

Die Analyse der Modelle wird eine Sammlung unterschiedlicher Konzepte zum Ergebnis haben, wie zum Beispiel das Tokenkonzept (das Herauslösung kleinster linguistischer Einheiten aus einem Primärdatum), das Spannenkonzept (die Zusammenfassung von Token in (dis-) kontinuierliche Spannen) oder auch das Hierarchiekonzept (die hierarchische Strukturierung von linguistischen Einheiten, wie ein Wort, als Teil eines Satzes etc.). Aufgrund dieser Sammlung kann ein Vergleich der Mächtigkeit der verschiedenen Modelle und ihrer konzeptuellen Überlappungen erfolgen.

3.2 (Meta-)Modell

Jedes der identifizierten Konzepte soll geeignet modelliert werden. Alle Konzepte werden anschließend in einem Metamodell zusammengeführt. Es muss explizit geprüft werden, inwiefern die vollständige Abdeckung der Konzepte der Quellmodelle durch das (Meta-)Modell erfolgreich war oder ob es bei der Transformation zu Informationsverlusten kommen muss.
Um das (Meta-)Modell für weitere Annotationsmodelle offen zu halten und eine einfache Übersetzung zu gewährleisten, soll das (Meta-)Modell möglichst frei von Semantik gehalten werden. Einschränkungen sollen nach Möglichkeit nur auf strukturelle Unterschiede und Überschneidungen der Konzepte hin modelliert werden. Dieser Ansatz führt dazu, dass im Modell Daten nicht aufgrund ihrer inhaltlichen Bedeutung repräsentiert werden, sondern aufgrund ihrer strukturellen Position. Damit kann der Verlust einiger Informationen (siehe Abschnitt 4) verbunden sein, die in den untersuchten Modellen vorhanden sind. Dieses Problem soll nach Möglichkeit mit der im nächsten Abschnitt dargestellten API behoben werden.
Durch die Semantikarmut des (Meta-)Modells ist die Semantik Interpretationssache der verarbeitenden Tools (dies könnten bspw. Im- bzw. Exporter sein). Um eine effiziente Parallelisierung zu erlauben, muss bereits hier darauf geachtet werden, dass unabhängige Partitionen entstehen - wie dies geschehen kann, ist im Laufe der Arbeit zu klären.

3.3 (Meta-)Modell -API

In der Arbeit wird eine API entwickelt, die das (Meta-)Modell verschiedenen Tools zugänglich macht. Ein solches Tool wird der zu entwickelnde Modellkonverter sein. Andere Tools könnten bspw. Editoren sein, sie sind jedoch nicht Teil dieser Arbeit. Die entwickelte API sollte einem großen Publikum von Entwicklern zur Verfügung stehen und daher nicht an eine Plattform gebunden sein.
Zur Modellierung des (Meta-)Modells und dessen Implementierung kann das `eCore`-Framework [4] verwendet werden. `eCore` ist eine Modellierungssprache und Teil des `Eclipse Modelling Frameworks` (`EMF`), [5]. Es existiert bereits eine Vielzahl von Technologien, die mit `eCore`-Modellen arbeiten können. So ist es möglich, aus einem `eCore` Modell eine Persistenzschicht, Model-2-Model Transformation und Editoren zu erzeugen.

3.4 Konverter

Für die Konvertierung der untersuchten Annotationsmodelle wird ein erweiterbares Framework entwickelt, das aus vier Komponenten besteht: den Importern, der Modell-API, den Exportern und einem Controller. Die Importer realisieren ein Mapping eines Quellformats auf das (Meta-)Modell und die Exporter das Mapping vom (Meta-)Modell auf das Zielformat. Bei der Konvertierung unterschiedlich mächtiger Formate lassen sich Informationsverluste nicht immer ausschließen. Über solche Fälle sollte der Nutzer informiert werden (siehe Abschnitt 4). Um den Konverter einem breiteren Publikum zugänglich zu machen, soll auch ein einfaches GUI entwickelt werden.

Performanz
Wie bereits erwähnt, spielt die Performanz des Konverter-Frameworks eine wichtige Rolle. Ein Ansatz zur Performanzsteigerung ist: die Parallelisierung der Konvertierung auf einem Cluster von Rechnern. Dafür muss eine geeignete Partitionierung des Konvertierungsprozesses gefunden und umgesetzt werden. Zur Realisierung der Verteilung wird untersucht, inwiefern sich der von `Google` entwickelte Ansatz `Map Reduce` [6] bzw. dessen Open Source Implementierung `Hadoop` [7] eignet.

Plugbarkeit
Aufgrund der kaum überschaubaren Anzahl existierender Modelle muss der Konverter erweiterbar sein. Daher soll ein Plug-In-Mechanismus entwickelt werden, der das Hinzufügen weiterer Im- und Exporter erlaubt. Für die in Abschnitt 3.1 definierten Formate werden im Rahmen dieser Arbeit je ein Im- und ein Exporter entwickelt, die ebenfalls auf Plug-In Basis in den Konverter integriert werden.

Individualisierbarkeit
Wie in Abschnitt 1 beschrieben, werden Korpora oft an das Tool bzw. das Modell oder Format, mit dem sie entwickelt wurden, angepasst. Informationen sind daher u.U. an falscher

Stelle platziert. Um dadurch entstehende Fehler bei der Konvertierung gering zu halten, wird untersucht, inwiefern es möglich ist, dem Nutzer die Möglichkeit zur Einflussnahme auf die konkrete Konvertierung zu bieten.

3.5 Optionale Erweiterungen

Merging
Eine besondere Herausforderung bei der Konvertierung von Daten stellt das Zusammenführen von mehreren Datenquellen in einen Korpus dar. Hier ist zu prüfen an welcher Stelle im Konvertierungsprozess eine Verschmelzung (Merging) stattfinden und anhand welcher Merkmale diese Verschmelzung vollzogen werden kann.

Modelltransformation
Um für die Zukunft die Entwicklung weiterer Konverter zu erleichtern kann geprüft werden, inwiefern es möglich ist ein Mapping eines Quell- bzw. Zielformats auf das (Meta-)Modell anzugeben, ohne dieses implementieren zu müssen. Hierzu könnte das Quell- bzw. Zielformat modelliert und anschließend versucht werden, über eine Model-2-Model Transformation ein Mapping automatisch zu erzeugen. Ein möglicher Ansatz könnte hier das `QVT`-Framework der `OMG` oder das `ecore2ecore` Framework von `EMF` darstellen.

4 Informationsverluste und Verlustfreiheit

Bei der Transformation von Daten von einem Modell auf ein anderes (Mapping) lässt sich der Verlust von Informationen prinzipiell nicht ausschließen. Mit jedem Mapping wächst die Gefahr Informationen zu verlieren. Daher versucht der hier vorgestellte Weg mit möglichst wenigen Schritten auszukommen. Zwar benötigt ein direktes 1:1-Mapping lediglich einen Schritt, während der Umweg über ein gemeinsames Modell zwei Schritte beinhaltet, der Vorteil des (Meta-)Modells wird aber bei einem Pipelining von Mappings deutlich. Aufgrund der Menge benötigter 1:1-Konverter existieren oft nur Wenige, die mit einem bestimmten Format umgehen können. Daher muss zu einer Hintereinanderschaltung von Konvertern gegriffen werden, die eine Vielzahl von Mappingschritten erfordern.
Unterstützt das Quellformat andere Konzepte als das Zielformat, muss es unweigerlich zu Informationsverlusten kommen. Um diese identifizierbar zu machen, müssen die Im- und Exporter über die vom entsprechenden Modell unterstützenden Konzepte Kenntnis haben. Diese können dann abgeglichen werden und der Nutzer soll vor der Konvertierung informiert werden, welche Informationen verloren gehen werden.
Da sich das entwickelte (Meta-)Modell nur mit der Struktur der untersuchten Modelle befasst, können auch nur strukturelle Informationsverluste bemerkt werden (bspw. ein Primärdatum kann nicht zu einem Token werden). Semantische Verluste (bspw. die Bedeutung einer bestimmten Annotation) bleiben Interpretationssache des verarbeitenden Tools. Werden bspw. Token zu einer „NP“ zusammengefasst, sollte diese Information bei Modellen, die Hierarchien unterstützen, nicht verloren gehen. Die einheitliche Bedeutung der Zeichenkette „NP“ kann

jedoch nicht sichergestellt werden (Semantikarmut).
Der Flaschenhals der Informationsverluste ist das (Meta-)Modell . Da es als Zwischenmodell dient, können nur Informationen exportiert werden, die im (Meta-)Modell darstellbar sind. Hier hilft lediglich eine möglichst genaue Untersuchung und Übersetzung der betrachteten Modelle.

5 Verwandte Arbeiten

Andere Ansätze zur Entwicklung eines übergreifenden Annotationsmodells sind PAULA [1] und das DDD -Modell [8]. Allerdings wurden diese Modelle nicht mit der gleichen Zielsetzung entwickelt wie sie in dieser Arbeit verfolgt wird. Beide Modelle/ Formate binden sich an bestimmte Technologien und lassen so nur eingeschränkt eine unabhängige Verarbeitung zu. Im Gegensatz dazu soll in dieser Arbeit ein abstraktes Modell entwickelt werden, das sich nicht auf eine bestimmte Technologie bezieht.

5.1 PAULA

PAULA wurde mit dem Ziel entwickelt, eine Persistenzierung mittels XML anzubieten, um Spannen- sowie Baumannotationen abbilden zu können. PAULA ist technologieabhängig, da es gezielt für XML entwickelt wurde. Eine dem PAULA-Modell entsprechende Hauptspeicherrepräsentation existiert (noch) nicht. Weiter fehlt eine vollständige formale Beschreibung des Modells, was eine Weiterverarbeitung erschwert. Aufgrund der Technologiegebundenheit ist der Annotator gezwungen, beim Arbeiten mit PAULA auf XML-Spezifika einzugehen und anhand dieser seine Korpusstruktur und Annotationen zu gestalten. Ein weiteres Problem ist, dass PAULA einige häufig anzutreffende linguistische Konzepte (wie gesprochene Korpora mit Zeitlinien und Verarbeitung mehrerer Texte in einem Dokument) nicht unterstützt.

5.2 DDD

DDD wurde mit dem Zweck entwickelt, ein Metasuchsystem für die Entwicklung eines Suchsystems für Korpora anzubieten. Dieses System sollte auf einer relationalen DB laufen. Zwar bietet DDD ebenfalls eine Persistenzierung in XML an, aber eine Hauptspeicherrepräsentation gibt es auch hier nicht. Ein weiteres Problem ist das Fehlen von Kantentypen und Kantenannotationen. Das führt dazu, dass Knoten nicht durch mehrere, inhaltlich verschiedene Kanten verbunden sein können. Dadurch sind Baum- und Spannenannotationen nur bedingt nutzbar.

Literatur

1. SFB 632 D1. PAULA: Interchange Format for Linguistic Annotations. http://www.sfb632.uni-potsdam.de/ d1/paula/doc/.
2. Department of Computational Linguistics, Institute of Natural Language Processing (IMS) in Stuttgart Phonetics in Saarbrücken, and the Institut für Germanistik in Potsdam. TIGER PROJECT - Linguistic Interpretation of a German Corpus. http://www.ims.uni-stuttgart.de/projekte/TIGER/.
3. Sonderforschungsbereichs Mehrsprachigkeit-(SFB 538). EXMARaLDA. http://www.exmaralda.org/.
4. Eclipse Foundation. Package org.eclipse.emf.ecore. Technical report, Eclipse Foundation, 2006.
5. Frank Budinsky, Dave Steinberg, Ed Merks, Ray Ellersick, and Timothy J. Grose. *Eclipse Modeling Framework (The Eclipse Series)*. Addison-Wesley Professional, August 2003.
6. Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. pages 137–150.
7. The Apache Software Foundation. HADOOP - Core. http://hadoop.apache.org/core/, 2007.
8. Thorsten Vitt. DDDquery: Anfragen an komplexe Korpora. Diplomarbeit, Humboldt Universitaet zu Berlin, Berlin, Germany, April 2005.