Präprozessierung mit Nebenwirkungen: Dynamische Annotation

Christoph Benden, Jürgen Hermes
Institut für Linguistik - Sprachliche Informationsverarbeitung
Universität zu Köln
Albertus-Magnus-Platz 1
50923 Köln
cbenden@spinfo.uni-koeln.de
and jhermes@spinfo.uni-koeln.de

Abstract

Maschinelle Korpusannotation hat zwei gravierende Nachteile: Zum Einen führt sie zu einer Verstetigung von Fehlern, zum Anderen erfordert sie eine unrevidierbare sprachtheoretische Entscheidung (letzteres gilt auch für manuelle Annotation). In diesem Paper wird ein Ansatz verfolgt, der die Annotation nicht primär als Element der zugrundeliegenden Datenstruktur - dem Korpus - sieht, sondern als Teilprozess der auf dem Korpus operierenden linguistischen Prozessketten. Dieser Ansatz manifestiert sich in einem erweiterten Präprozessor (Extended Preprocessor - XPre), dessen auf Plug-ins beruhende Architektur es erlaubt, Daten einerseits theoriespezifisch auszuzeichnen, andererseits aber beliebig modifizierbar zu halten. Statt theorieneutrale (und damit oft unangemessene) Tagsets einmalig zuzuweisen, sind theoriespezifische Tagsets aufgabenbezogen anwendbar. Verbessernde oder erweiternde Eingriffe in den Annotationsmechanismus stehen dem Anwender genauso offen, wie die Auswahl und Entwicklung beliebiger Tokenisierungs- und Akkumulationsprozesse.

1 Statische Korpusannotation vs. prozessorientierte Annotierung

Die Annotation von Sprachdaten in einem Korpus entspricht einer Anreicherung der zugrundeliegenden Datenstruktur (Text) durch explizite Auszeichnungselemente (Tags), mit deren Hilfe implizite Information ausgedrückt werden soll (vgl. McEnery (2003, 453)). Annotation im Korpus wird allgemein als zweckmäßig angesehen, da hierdurch linguistische Analysen aufgezeichnet und damit wiederverwendbar werden. Nutzer des Korpus können so auf Analysen zurückgreifen, die sie selbst nicht oder nur unter hohem Zeitaufwand hätten erstellen können.

Obwohl von impliziter Information ausgegangen wird, muss diese erst kenntlich gemacht werden, was von einem Linguisten, maschinell oder durch menschlich-maschinelle Zusammenarbeit geleistet werden muss. Für die maschinelle Auszeichnung stehen inzwischen eine Reihe von Tools zur Verfügung, mit denen unterschiedliche Einheiten des Textes (Buchstaben, Wörter, Sätze, Paragraphen, gesamte Texte) gelabelt werden können. Problematisch dabei ist, dass die Anzahl der Fehler automatischer Annotation auch in schon relativ gut ausgearbeiteten Bereichen wie der Wortarten-Labelung (Part-of-speech) und der Lemmatisierung immer noch so hoch ist (bis zu 3%, vgl. McEnery (2003, 455)), dass sie nicht ohne weiteres akzeptiert werden kann. Menschliche Nachbearbeitung oder gar ausschließlich manuelle Auszeichnung umfangreicher Korpora sind für kleinere Forschungsvorhaben mit begrenzeten Ressourcen jedoch nicht zu leisten.[1]

Wird die maschinell erzeugte fehlerhafte Information nun als Annotation auf die zugrundeliegenden Sprachdaten angewendet, so verstetigen sich die Fehler des Annotationsalgorithmus in der Datenstruktur. Wird der Algorithmus verbessert, muss die gesamte Datenstruktur (d.h. das gesamte Korpus) der neuen Entwicklung angepasst werden.

Ein weiteres Problem für das Annotieren von Texten ergibt sich aus der lediglich subjektiven Interpretation der Daten. Zwar werden Korpusannotatoren dazu angehalten, die Daten möglichst theorieneutral auszuzeichnen (vgl. Leech (1993, 275ff)); indess kann es keine echte Theorieneutralität geben. Jede Auszeichnung

[1] Darüber hinaus wird an der Konsistenz von manuell ausgezeichneten Sprachdaten gezweifelt (vgl. Sinclair (1992)).

ist auch immer eine Interpretation hinsichtlich eines zugrundegelegten Modells über sprachliche Strukturen. Dies zeigt sich in den verschiedenen Tagsets für die Auszeichnung von Wortarten, welche die Grundlage für gebräuchliche Tagger sind bzw. mit denen gebräuchliche große Korpora ausgezeichnet werden. Grundlage für die meisten dieser Tagsets bilden die zehn klassischen Wortarten (Nomen, Verben, Adjektive, Präpositionen usw.), deren Ermittlung in sich nicht konsistent ist, da sowohl morphologische Eigenschaften als auch das syntaktisch-distributive Verhalten der Wörter zur Klassifikation herangezogen werden. Daneben wird ausgeklammert, dass man die Klassifizierung von Wörtern auch auf andere Arten vollziehen kann, etwa durch reine Distributionsanalyse. Es kann kein Tagset existieren, welches alle möglichen Wortartenklassifikationen in sich vereint und nicht zugleich vollkommen unübersichtlich und inkonsistent wird. Annotation bedeutet damit auch immer Entscheidung hinsichtlich der theoretischen Grundlagen der weiteren Sprachverarbeitung.

Wie sich andeutet, hat die Annotation, soweit man sie auf ein bestehendes Korpus anwendet, zumindest zwei gravierende Nachteile:

1. Korpusannotation führt zu einer Verstetigung von Fehlern bzw. Inkonsistenzen oder beidem

2. Korpusannotation bedingt eine unrevidierbare (sprach)theoretische Entscheidung

Grundidee unseres Vorschlages zur Lösung der Annotationsnachteile ist die Nutzung eines Korpus mit relativ flacher Struktur, in dem nur absolut sicher identifizierbare Elemente ausgezeichnet sind. Im Rahmen des von der Fritz Thyssen Stiftung geförderten SemGen-Projekts wurde von Mitarbeitern der Sprachliche Informationsverarbeitung ein Korpus auf der Basis von rund 330.000 Zeitungsartikeln erstellt, welches 98,2 Millionen Token als Instanzen von 2,1 Millionen Types umfasst. Die Texte wurden in ein XML-konformes Format gebracht und in eine native XML-Datenbank (Tamino) eingespeist.

Die Metainformationen des Zeitungskorpus beschränken sich auf die Separierung nach Texten, die wiederum in Paragraphen unterteilt sind.[2] Textauszeichnungen enthalten daneben noch außertextuelle Informationen über Typ, Quelle, Erscheinungsdatum, Sprache und Sparte.

Auf dieser Korpusstruktur setzt der Präprozessor auf, der - im Vergleich zu herkömmlichen Präprozessoren - kein simpler Tokenizer ist, sondern zudem Akkumulations-, Klassifizierungs-, und Annotationsfunktionalitäten umfasst. Diese Erweiterung eines einfachen Präprozessors schlägt sich auch in seinem Namen - XPre für e*X*tended *Pre*processor - nieder. Diese Applikation ist einerseits dazu in der Lage, Texte unterschiedlichsten Formats einzulesen, auf unterster Ebene - der Zeichenebene - zu separieren, separierte Zeichen wiederum in größeren Einheiten (Wort, Numeral, Satz etc.) zu akkumulieren und diese Einheiten schließlich mit (linguistischer) Information auszuzeichnen. Durch diese Verfahrensweise wird man vor der Verstetigung von Fehlern, dem ersten oben angebrachten Nachteil der Korpusannotation, bewahrt: Die Annotation ist Teil eines Prozesses; Änderungen des Annotationsalgorithmus, z.B. die Verbesserung der Satzgrenzenerkennung oder eine Weiterentwicklung des POS-Taggers, haben so keine direkte Auswirkung auf die bestehende Datenstruktur (das Korpus), da immer nur ausgewählte Ausschnitte prozessiert und damit annotiert werden.[3]

Auch der zweite oben genannte Nachteil wird durch die Architektur des XPre ausgeräumt: Die Annotationskomponente ist vollständig durch Plug-ins realisiert, die dem Anwender freistellen, auf welches Tagset er zurückgreifen will (zum Einen stehen vorgefertigte Tagsets bereit, zum Anderen ist es möglich, eigene zu entwerfen) und welche Art der Klassifikation er überhaupt anwenden will (bspw. klassische vs.

[2] Unter dem <`Paragraph`>-Tag sind alle durch einen Zeilenumbruch abgeschlossenen Textabschnitte zusammengefasst, also auch Überschriften. Um letztere mit einem eigenen Tag zu labeln, wäre eine Heuristik notwendig gewesen, die Überschriften von anderen Paragraphen unterscheiden könnte. Eine solche Heuristik arbeitet nicht vollkommen präzise und hat deswegen ihren Platz im unten beschriebenen Präprozessor.

[3] Es bleibt hierbei dem Anwender überlassen, welchen Umfang seine zur Prozessierung ausgewählten Korpusausschnitte haben. Es ist auch möglich, das gesamte Korpus zu selegieren.

distributionell ermittelte Wortarten). Der Philosophie nur scheinbar theorieneutraler Tagsets setzt XPre die Möglichkeit theoriespezifischer Annotation für jeweils unterschiedliche Aufgaben entgegen.

2 XPre: Ein erweiterbarer, annotierender und persistierender Präprozessor

Das Programmprojekt XPre bietet eine hochflexible Applikation zur Präprozessierung und Annotation von Korpora, die z. Zt. in zwei Szenarien eingesetzt wird:

1. als präprozessierende Komponente in SEMALD, einem System zur Erstellung von Prozessketten zur Verarbeitung geschriebener Korpora bzw. Korpusausschnitten und

2. als Stand-alone-Anwendung zur Segmentierung, Klassifizierung und Annotation von Korpustexten, entweder als Echtzeitanwendung oder mit dem Ziel der Persistierung des Ergebnisses.

Im Kern besteht XPre aus einer Verwaltungsstruktur von geschichteten Repäsentationsebenen und Algorithmen (Parsern) zur jeweiligen Überführung einer Repräsentationsebene in die nächsthöhere. In der gegenwärtigen Version von XPre werden vier Repäsentationsebenen realisiert (vgl. Abbildung 1), prinzipiell sind aber weitere Ebenen (etwa ein `TextLayer`) möglich.[4] Dem Anwender steht offen zu wählen, bis zu welcher Ebene die Eingabezeichenkette analysiert werden soll.

Der Kern des Präprozessors bietet Schnittstellen zu Peripherie-Modulen (Plug-ins), welche die Ebenen, auf denen sie applizieren, mit zusätzlichen Annotationen versehen. Dem Anwender steht frei, bestehende Plug-ins auszuwählen, als auch eigene Plug-ins zu entwickeln und einzubinden.

Die Auswahl der Funktionalität von XPre wird anhand einer XML-Datei bestimmt, die dem

[4]Wir haben uns aus Übersichtsgründen dafür entschieden, den Ebenen den Namen herkömmlicher linguistischer Entitäten zu geben. Streng genommen müssten wir von ZeroOrderLevel, FirstOrderLevel etc. spechen, da auf dem WordLevel eben nicht nur Wörter aus Zeichen zusammengesetzt werden, sondern auch Interpuntionszeichen disambiguiert werden (s.u.).

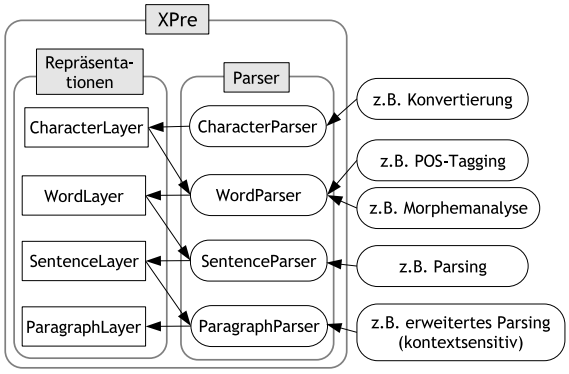

Abbildung 1: XPre - Architektur

```
<Settings>
  <CharacterParserSettings>
    <ParserSettings>
      <Char>  <Item>a</Item> ... <Item>Ü</Item> </Char>
      <Digit> <Item>1</Item> ... <Item>0</Item> </Digit>
      <Dot>   <Item>.</Item> </Dot>
    </ParserSettings>
    </AnnotatorSettings>
  </CharacterParserSettings>
  <WordParserSettings>
    <ParserSettings>
      <Pattern>
        <StartsWith>Char</StartsWith>
        <Contains>Char</Contains>
        <Contains>Digit</Contains>
        <!-- Man denke an E-Mail-Adr., URLs etc. -->
        <Contains>Dot</Contains>
        <Ambiguity>
          <LayerElement>Dot</LayerElement>
          <Resolver>DotResolver</Resolver>
        </Ambiguity>
      </Pattern>
    </ParserSettings>
    <AnnotatorSettings>
      <Annotator>POSTagger</Annotator>
      <Annotator>DistributionalClassifier</Annotator>
    </AnnotatorSettings>
  </WordParserSettings>
  <SentenceParserSettings>
    <ParserSettings>
      <Pattern>
        <StartsWith>Word</StartsWith>
        <StartsWith>Numerical</StartsWith>
        <Contains>Word</Contains>
        <Contains>Numerical</Contains>
        <!-- Umfaßt hier , ; : - etc. -->
        <Contains>Punctuation</Contains>
        <!-- Umfaßt hier ? ! . -->
        <EndsWith>FullStop</EndsWith>
      </Pattern>
    </ParserSettings>
    </Annotator>
  </SentenceParserSettings>
</Settings>
```

Abbildung 2: XPre - Settings

Format in Abbildung 2 entspricht (hervorgehobene Tags entsprechen reservierten Bezeichnern; andere Bezeichner sind frei wählbar).

Die einzige Ebene, die direkt auf den Daten des zu prozessierenden Objekts operiert, ist der `CharacterParser` (CP), der jedes Zeichen des Texts entsprechend der Vorgaben in `<Item>` als `Char`, `Digit`, `Dot` etc. klassifiziert und im `CharacterLayer` (CL) ablegt. Sowohl die Kategoriennamen als auch die `<Item>`-Definitionen

sind frei wählbar.

Auf allen höheren Ebenen werden die Objekte der jeweils darunter liegenden Ebene entsprechend der Definitionen in <StartsWith> + <Contains> bzw. <StartsWith> + <EndsWith> akkumuliert. Der WordParser (WP) akkumuliert bspw. die Elemente des CL zu Word-Objekten, die mit einem Char beginnen und ausschließlich aus Char-, Digit- und Dot-Objekten (im Beispiel) bestehen und legt sie im WordLayer (WL) ab. Der SentenceParser (SP) wiederum akkumuliert WL-Objekte zu Objekten des SentenceLayer (SL), die im Beispiel mit einem Word- oder Numeral-Objekt beginnen, Word-, Numeral- oder Punctuation-Objekte enthalten und mit einem FullStop-Objekt[5] enden.

Als Beispiel für ambige Zeichen wird hier der '.' betrachtet, der Satzenden, Abkürzungen, Domänennamen etc. markieren bzw. diskriminieren kann. Auf dem CL wird der Punkt neutral als Dot klassifiziert, da hier keine Disambiguierung vorgenommen werden kann. Auf dem WL wird die Ambiguität mittels einer Klasse (hier: DotResolver) aufgelöst, die entweder mitgeliefert wird oder, bei besonderen Ansprüchen, anhand des Interface Resolver selbst ausprogrammierbar ist.[6]

Mittels der <Annotator>-Deklaration wird im Beispiel der WP angewiesen, die beiden annotierenden Komponenten POSTagger und DistributionalClassifier[7] einzubinden. Dies erfolgt mittels eines (minimalen) Komponentenmodells, welches die Komponenten - wenn erforderlich - initialisiert, ausführt und die Resultate in den zugehörigen Layer-Objekten speichert. Diese Möglichkeit der Einbindung beliebiger Komponenten[8] und der Zwischenspeicherung ihrer Resultate soll auf Dauer eine ausschließlich persistente Annotation durch algorithmisch beschriebene Klassifizierungen ersetzen, die erst während der Präprozessierungsphase, d.h. unmittelbar nach dem Korpuszugriff, applizieren.

Für den Benutzer bestehen zwei Zugriffsmöglichkeiten auf die Resultate von XPre: Entweder können die Daten direkt weiterverarbeitet werden oder über eine bereitgestellte Schnittstelle in einer beliebigen Datenbank persistiert werden.

3 Fazit

Sicherlich kann man in unserem Ansatz zur bedarfsorientierten Annotation bemängeln, dass eine korrigierende oder erweiternde Auszeichnung von Hand nicht vorgesehen ist. Allerdings lassen sich Ergebnisse von XPre selegieren und für nachträgliche Bearbeitung zugänglich machen. Die bearbeiteten Daten lassen sich ohne weiteres persistieren, wodurch aber natürlich der gewichtigste Vorteil unserer dynamischen Annotation - das Einsparen gewaltiger Verwaltungs- Speicher- und Verweisstrukturen - teilweise zunichte gemacht wird.

References

C. Benden. erscheint. Automated detection of morphemes using distributional measurements. In *Proceedings of the 28. Annual Conference of the GfKl in Dortmund, March 9th-11th, 2004*. Springer, Berlin.

G. Leech. 1993. Corpus annotation schemes. *Literacy and Linguistic Computing*, 8(4):275–81.

T. McEnery. 2003. Corpus linguistics. In R. Mitkov, editor, *The Oxford Handbook of Computational Linguistics*, S. 448–463. University Press, Oxford.

J. Sinclair. 1992. The automatic analysis of corpora. In J. Svartvik, editor, *Directions in Corpus Linguistics*, S. 379–397. Mouton de Gruyter, Berlin, New York.

[5] FullStop- und Punctuation-Objekte sind hierarchisch gegliederte Objekte, die wiederum weitere (hierarchisch gegliederte) Unterobjekte enthalten. Solche Objekte sind für jeden Layer definierbar.

[6] Die deklarative Definition ambiger Kontexte innerhalb der Steuerungsdatei, die eine explizite Ausprogrammierung der Auflösung von Ambiguitäten vermeidet, ist für die nächste XPre-Version projektiert.

[7] Hierbei handelt es sich um zwei Komponenten, die ursprünglich für SEMALD entwickelt wurden, einem Tagger, der auf ein Vollformlexikon zugreift sowie eine Morphemanalyse, die auf einer distributionellen Auswertung von Graphemsequenzen beruht, vgl. Benden (erscheint).

[8] Die Komponenten müssen natürlich auf ein bestimmtes Interface hin programmiert sein, da bereits vorhandene Komponenten entsprechende Wrapper benötigen. Siehe Dokumentation unter http://www.spinfo.uni-koeln.de/forschung/papers/xpre/manual.html.