Standoff-Tools – Generische Dienste für die automatische Annotation von XML-Dokumenten mit Plain-Text-Werkzeugen

Lück, Christian
christian.lueck@uni-muenster.de
Westfälische Wilhelmsuniversität Münster, Deutschland

TEI-XML ist eine exzellente Technologie für digitale Editionen. Aber für die computationelle Analyse ist ein Korpus von XML-Dokumenten keine günstige Grundlage, weil Algorithmen erheblich komplexer werden, wenn statt linearem plain text ein XML-Baum durchlaufen werden muss. Zwar ist die Extraktion von plain text aus XML nicht aufwendig, aber für die Rückführung von Analyse-Ergebnissen in den XML-Baum gibt es bislang keine allgemeine Lösung. Dennoch ist eine solche in vielen Fällen wünschenswert, weil dann Analyse-Ergebnisse und die Struktur des Dokuments in einer allgemeinen Abfragesprache wie XQuery ausgewertet werden können. Es existieren Insellösungen zum Enrichment von TEI-XML, die auf bestimmte Tools ausgerichtet sind, etwa Spacy (Andorfer und Schlögl 2021, Meyer 2022), oder auf bestimmte Anwendungsfälle, etwa Alliterationen (Consalvi und Fumagalli 2022). Wünschenswert wäre jedoch, ganz allgemein Analyse-Tools für plain text entwickeln zu können und sie auch zum Enrichment von XML einsetzen zu können. Genau dies ermöglichen die hier vorgestellten *StandOff Tools*.

Komponenten für Annotationspipelines

Die *StandOff Tools* bestehen aus zwei Komponenten:[1] dem Extraktor *E* und dem Internalizer *I*. In einer Annotationspipeline ist zwischen *E* und *I* ein anwendungsspezifischer Plain-Text-Tagger *T* geschaltet. (Abb. 1) *E* extrahiert aus dem XML-Quelldokument plain text, der an *T* geleitet wird. *I* nimmt von *T* gelieferte Referenzierungen von Plain-Text-Fragmenten entgegen und bringt sie als Inline-Markup stets so in den XML-Baum ein, dass wohlgeformtes XML herauskommt.

E und *I* sind insofern generische Komponenten, als dass ein beliebiger Tagger in eine Pipeline zum Enrichment von XML eingesetzt werden kann, solange er folgende Anforderungen erfüllt: a) Er muss plain text verarbeiten, b) er muss darin Textpassagen (ranges) per *character offsets* referenzieren, genauer: nach den in RFC 5147, Abschnitt 2.1.1 und 2.2.2 beschriebenen Regeln für *character ranges*.[2] Die vom Tagger ausgegebenen Ranges dürfen einander umfassen oder überlappen, so dass auch komplexe Strukturen in der XML-Quelle ausgezeichnet werden können. Die Ranges werden von *I* so zerschnitten, dass Überlappungen mit anderen Ranges und dem internen Markup der XML-Quelle aufgelöst werden (Splitting).

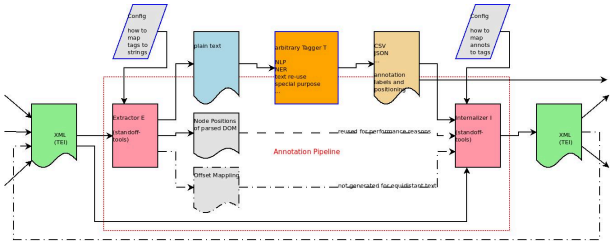

Abb. 1: Schema einer Annotationspipeline für TEI XML

Ansätze

Die Komponenten *E* und *I* müssen so aufeinander abgestimmt sein, dass *I* hinreichend Informationen hat, um die Annotationen von *T* in das XML-Dokument einzubauen. Neben Ansätzen, die ein Sprachmodell, und sei es auch nur eine Tokenisierung, einführen und insofern nicht generisch sind (Schopper 2021, Andorfer und Schlögel 2021), lassen sich in der DH-Software-Entwicklung zwei Ansätze beobachten.

Nach dem ersten extrahiert *E* die Textknoten und sammelt dabei Informationen darüber, wo Elemente anfangen und aufhören. Dies ist auf DOM-Ebene möglich. *I* kann dann Blätter des DOM-Baums so ersetzen, dass an ihrer Stelle neue Teilbäume die (zerschnittenen) Annotationen darstellen. Diesen Ansatz verfolgen die Lösung von Meyer (2022) und der Prototyp von Lassner (2021). Sein Vorteil ist, dass gängige XML-Parser eingesetzt werden können. Sein Nachteil ist, dass manche Merkmale der serialisierten XML-Quelle nicht bewahrt werden können, weil sie nicht zur DOM-Spezifikation gehören: *character* und *entity references* sowie *whitespace* in Tags.

Die *StandOff Tools* verfolgen einen anderen Ansatz: Hier speichert *E* Informationen über die Position, die jedes Zeichen des extrahierten Textes im XML-Quelldokument gehabt hat (*offset mapping*). DOM-Manipulationen erfolgen nicht. Ein Parser ermittelt Positionsdaten aus dem XML-Quelldokument. Die Extraktion erfolgt dann im Wesentlichen durch Kopieren von Zeichenketten aus dem serialisiert vorliegenden XML-Quelldokument. *I* kann aufgrund des *offset mapping* die Positionsdaten des Taggers auf die XML-Quelle beziehen. Wie im ersten Ansatz zerschneidet *I* die Annotationen. Das Splitting, das Herzstück des Algorithmus, basiert auf einer Auswertung der Positionsdaten (offsets), die der XML-Parser für das interne Markup und der Tagger *T* für die Annotationen liefert. Anschließend werden aufgrund der Positionsdaten Portionen der XML-Quelle in die Ausgabe kopiert und neue Tags dazwischen gesetzt. Dieser Ansatz hat den Nachteil, dass kein üblicher XML-Parser eingesetzt werden kann. Aber er reproduziert diejenigen

Merkmale der XML-Quelle, welche im ersten Ansatz verloren gehen.

Einsatz für manuelle Annotationen

I kann auch allein betrieben werden, um manuelle Annotationen, die Passagen eines XML-Dokuments per character offsets referenzieren, zu internalisieren und zu anschließend zu visualisieren oder auszuwerten. Solche Annotationen können z.B. als Web Annotations (OA-Ontology) realisiert werden. Wenn dabei allerdings wie in CATMA eine schlichte Extraktion von Textknoten aus HTML oder XML erfolgt, gehen dabei Informationen verloren, welche für eine Internalisierung der Annotationen erforderlich sind. Insofern bleibt die Internalisierung von CATMA-Annotationen in TEI noch Desiderat (Cayless 2019). Dennoch weisen die verschiedenen neuen Entwicklungsansätze einen Weg abseits vom nicht-funktionalen Standoff-Konzept der TEI (Banski 2010), auf welchem der Mehrwert sowohl von Standoff-Annotationen als auch von TEI realisierbar ist.

Fußnoten

1. Die Komponenten liegen derzeit als Kommandozeilen-Programme vor. Eine Implementierung als Webservices ist in Planung, wobei sich die Modellierung des Informationsflusses komplexer gestaltet, weil der Extraktor einen mehrfachen Output hat.
2. Derzeit steht als Eingabe-Format für den Internalizer CSV zur Verfügung. Eine Implementierung der Syntax von RFC 5147 ist geplant.

Bibliographie

Andorfer, Peter und Schlögl, Matthias. 2021. acdh-spacytei. In: KONDE Weißbuch. Hrsg. v. Helmut W. Klug unter Mitarbeit von Selina Galka und Elisabeth Steiner im HRSM Projekt "Kompetenznetzwerk Digitale Edition". Aufgerufen am: 3.8.2022. Handle: hdl.handle.net/11471/562.50.2.

Banski, Piotr. 2010. "Why TEI stand-off annotation doesn't quite work and why you might want to use it nevertheless." Balisage: The Markup Conference 2010, 10.4242/BalisageVol5.Banski01

Cayless, Hugh. 2019. "Implementing TEI Standoff Annotation in the browser." Proceedings of Balisage: The Markup Conference 2019, no. 23, 10.4242/BalisageVol23.Cayless01

Consalvi, A. und Fumagalli, S. 2022. "Alliteration: automatic identification and encoding." 2022 TEI Conference. Conference abstract, Session 1A-2, (im Erscheinen)

Lassner, David. "The Standoff Converter. A standoff-based approach to work on TEI documents in Python that connects the world of digital philology with NLP." 2021 TEI Conference and Members' Meeting.

Meier, Wolfgang. 2022. "Names sell. Named Entity Recognition in TEI Publisher", https://e-editiones.org/blog/ (zugegriffen: 3. August 2022)

Schopper, Daniel. 2021. xsl-tokenizer. In: KONDE Weißbuch. Hrsg. v. Helmut W. Klug unter Mitarbeit von Selina Galka und Elisabeth Steiner im HRSM Projekt "Kompetenznetzwerk Digitale Edition". Aufgerufen am: 3.8.2022. Handle: hdl.handle.net/11471/562.50.216.