StandOff Tools

Christian Lück
Service Center for Digital Humanities
Westfälische Wilhelms-Universität Münster

Idee

Wiederkehrende Herausforderung

- TEI-XML gute Technologie für digitale Editionen
- aber nicht für computationelle Analysen; dort Plain-Text besser geeignet
- Plain-Text zwar einfach extrahierbar, aber:
- Rückführung von Analyse-Ergebnissen in TEI-XML ungelöste Aufgabe

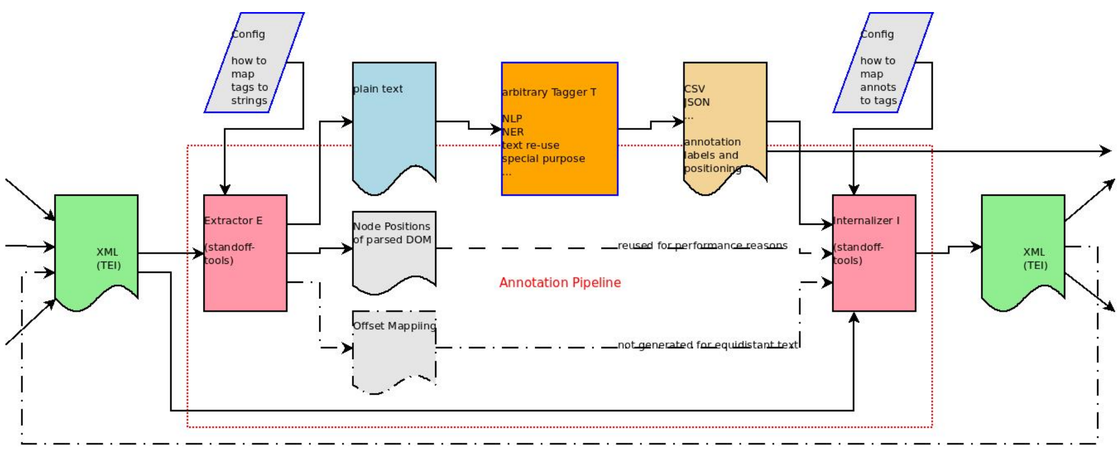
Ziel

Generische Werkzeuge für XML-Annotations-Pipelines mit Taggern für Plain-Text

Designprinzipien

- keine Trennzeichen (space) voraussetzen, vgl. scriptura continua
- kein Sprachmodell voraussetzen (Sätze, Wörter, Morpheme, etc.)
- Referenzierung per Character-Offsets stets eindeutig, auch bei repetitivem Inhalt
- weitgehende Abstraktion von XML, geeignet für andere hierarchische ML

StandOff Tools



- zwei aufeinander abgestimmte Komponenten: Extraktor und Internalizer
- kein Informationsverlust wie bei Extraktion mit gewöhnlichen X-Technologien

Extraktor

Aufgabe: Herstellung von Plain-Text unter Wahrung der eindeutigen Referenzierbarkeit bzgl. des Quell-Dokuments

Lösung: Reihenfolge-stabile Plain-Text-Extraktion mit *zugehörigem Offset-Mapping* (siehe rechts)

Beispiele:

- Text ohne Metadaten, Prätexte usw.
- Text ohne Anmerkungen, krit. Apparat usw.
- nur ausgewählte Teile eines Textes, z.B. Sprechanteile einer dramat. Figur

Internalizer

Aufgabe: Anreicherung des Quell-Dokuments mit Inline-Markup unter Wahrung der Wohlgeformtheit

Lösung: Splitting der vom Tagger bestimmten Text-Passagen, so dass zum Markup im Quell-Dokument passend

Beispiel: Ein Tagger findet die blaue Passage, welche Vers-Grenzen überschreitet:

```
… Scythiam septemque trionem
Horrifer invasit Boreas. …
```

```
<l>… Scythiam <t id="s1">septemque trionem</t></l><t id="s2" prv="#s1">
</t><l><t id="s3" prv="#s2">Horrifer</t> invasit Boreas. …</l>
```

- internes Markup (violett) des Quell-Dokuments bleibt unverändert
- neues Markup (orange, rot) wird gesplittet und aggregiert (`@prv`)
- neues Markup ist wohlgeformt, aber u. U. nicht valide (rot)

Plain-Text-Extraktion und Referenzierung

Quell-Dokument mit Markup

```
<d><m>Metamorphoses</m><t>In nova fert animus mutatas dicere …</t></d>
    ^    ^    ^    ^    ^    ^    ^    ^    ^    ^    ^    ^    ^
         10        20        30        40        50        60        70
```

XML-Parser liefert DOM-ähnliche Repräsentation mit Positionsinformationen zu jedem Knoten. Hier in Lisp-Schreibweise:

```
(elem "d" 1 70 '((syntax 1 3) (syntax 67 70))
  '((elem "m" 4 23 '((syntax 4 6) (syntax 20 23))
      '((text 7 19 "Metamorphoses")))
    (elem "t" 24 66 '((syntax 24 26) (syntax 63 66))
      '((text 27 62 "In nova …")))))
```

Äquidistante Plain-Text-Extraktion

```
______Metamorphoses_____In nova fert animus mutatas dicere …_______
```

Syntaktische Elemente der ML werden durch gleich lange Zeichenketten ersetzt. Jedes Zeichen eines Textknotens hat denselben Offset wie im Quell-Dokument.

⇒ Offset-Referenzierung unmittelbar auf Quell-Dokument übertragbar

Geschrumpfte Plain-Text-Extraktion

```
Metamorphoses_In nova fert animus mutatas dicere …
```

Syntaktische Elemente der ML werden aufgrund einer Ersetzungstabelle ersetzt, wahlweise durch die leere Zeichenkette. Hier: `{close-m: "_", *: ""}`

⇒ Offset-Referenzierung übertragbar auf das Quell-Dokument mittels Offset-Mapping: 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 27, 28, 29, …

Geschrumpfter Plain-Text und *zugehöriges* Offset-Mapping lassen sich aus dem DOM mit Positionsdaten herstellen.

Reihenfolge-stabile Transformation

```
(elem "d" 1 70 '((syntax 1 3) (syntax 67 70))
  '((elem "m" 4 23 '((syntax 4 6) (syntax 20 23))
      '((elem "deleted" 7 19 '((syntax 7 19)) '()))
    (elem "t" 24 66 '((syntax 24 26) (syntax 63 66))
      '((text 27 62 "In nova …")))))
```

Beliebige (Teile von) Text-Knoten dürfen durch leere Element-Knoten ersetzt werden. Dies entspricht der Identitätstransformation in XSLT erweitert um einen eingeschränkten Satz von Regeln für Textknoten.

Es ist nur die DOM-Repräsentation vorgesehen. Eine Serialisierung wäre:

```
<d><m><deleted/></m><t>In nova fert animus mutatas dicere …</t></d>
```

Reihenfolge-stabile Plain-Text-Extraktion

```
In nova fert animus mutatas dicere …
```

Das ist eine funktionale Komposition einer reihenfolge-stabilen Transformation und einer geschrumpften Plain-Text-Extraktion.

⇒ Offset-Referenzierung übertragbar auf das Quell-Dokument mittels Offset-Mapping: 27, 28, 29, …

⇒ **geeignetes Ziel-Format für Tagger**

Grenzen liegen bei Änderungen in der Textreihenfolge oder Ersetzungen, z.B. Herstellung des Textes einer Variante bei externem krit. Apparat.

Anwendungen

- spaCy als Tagger in Pipeline
- CiteRefParser als Tagger in Pipeline: automatische Auszeichnung von Bibel-Referenzen