XPath- und XSLT-Übungsblatt (LV-WS20/21, *Textkodierung und Textanalyse mit TEI*)

Sarah Lang

Januar 2021

XPath-Übungsblatt für die Lehrveranstaltung 'Textkodierung und Textanalyse mit TEI, UE' (Universität Wien).

Block 1: C. Pollin. (Intro to XML)
Block 2: R. Bleier. (TEI)
Block 3: H. Clausen. (Text, Materialität, fortgeschrittene TEI)

Block 4: Fr, 15.01.2021 – Publikationslösungen, XPath, XSLT.

Nutzen Sie die XML/TEI-Dateien, die Sie zur Übung und zum Erlernen von XML/TEI erstellt haben als Basis für die folgende Übung. Interessantere Ergebnisse kommen normalerweise heraus, wenn das Dokument nicht allzu klein ist und doch einige Annotationen vorhanden sind. Öffnen Sie das Dokument im XML-Editor Ihrer Wahl.

1. Verwenden Sie das XPath-Feld für die XPath-Übung.
2. Erstellen Sie ein `.xsl`-Stylesheet mithilfe der Informationen aus Video/Folien oder Cheatsheet (dort ist das sehr genau erklärt).
3. Stellen Sie gleich als erstes ein Transformationsszenario ein (siehe Video dazu sowie Cheatsheet, Abschnitt 9 'Einstieg').
4. Fahren Sie nicht mit den Übungen fort, bevor Sie das getan haben und es funktioniert.

Geben Sie Ihre Lösungen auf Moodle ab – Ihre Notizen als digitaler Text, Foto/PDF/Scan Ihrer handschriftlichen Notzizen oder von mir aus Zeichnungen. Alles ist erlaubt, solange es einigermaßen lesbar für mich ist und ich die Dateien öffnen kann!

Bitte auch eine Liste der XPath-Abfragen abgeben und ein fertiges XSL-Dokument. Wenn nicht alles geht, ist es nicht dramatisch, aber bitte erkennbare Versuche, sonst keine Punkte!

Fragen dürfen jederzeit gestellt werden.

1 XPath-Übungen

1.1 Formale Kriterien abfragen

1. Kapitel/Abschnitte zählen.
2. Überschriften ausgeben lassen.
3. Mit `substring(), substring-before(), substring-after()` Kapitel-Heads ausgeben lassen, die einen bestimmten Text enthält.
4. Mit `distinct-values()` alle vorkommenden Personennamen genau 1x ausgeben lassen.
5. Vorkommende Namensvarianten printen (falls vorhanden, evtl. auch `distinct-values()`).
6. Mit lokalen und globalen *scopes* herumprobieren (absolute und relative Pfade ausprobieren). Was sind die Vor- und Nachteile? Wann würde ich welches verwenden?
7. `matches()` und `contains()` ausprobieren: Achtung, `matches()` kann RegEx, `contains()` nicht!
8. Die Überschrift des 4. Teils.
9. Personenname, der nicht 'peter' enthält.
10. Alle Personen oder Orte eines Abschnitts (mit einer Abfrage zu printen).

1.2 'Statistiken' erheben und Metadaten abfragen

1. Wie viele Personen/Orte/etc. gibt es?
2. Welche Orte/Leute kommen (wann und wo) vor?
3. Wer hat den Text geschrieben?
4. Welche Informationen (Metadaten9 lassen sich aus dem TEI-Header erheben und wie?
5. Welche Personen sind zusammen in Abschnitten? Wer 'trifft' sich nie?
6. Welche Personen kommen mit welchen Orten zusammen vor (falls vorhanden?

2 XSLT

2.1 Vorbereitung der Transformation

1. Welche Attribute und Elemente kommen vor? (selbstständig googeln, wie man das machen kann oder das Cheatsheet diesbezüglich konsultieren).
2. Welche Attributwerte wurden vergeben?
3. Aufschreiben / Liste machen und überlegen, welche davon man im Zuge einer Webdarstellung erhalten/darstellen möchte.
4. Googeln, wie diese in HTML heißen oder dargestellt werden könnten. (Achtung, im Gegensatz zu TEI ist HTML nicht nach Semantik, sondern nach Darstellungskategorien organisiert!)
5. Welche Zusatzinformationen könnten Nutzer:innen meiner digitalen Edition interessieren? Statistiken?

2.2 Erarbeiten von XPath-Abfragen für die Template-Regeln

1. Die zu verarbeitenden Elemente notieren: Brauche ich Push- oder Pull-Paradigma? D.h. wo in meinem Stylesheet muss die Abfrage hin? Möchte ich die Elemente verarbeiten/erhalten, wo sie auftauchen oder möchte ich entscheiden, wo sie hingesetzt werden?
2. Brauche ich absolute oder relative Pfade?
3. In natursprachlichen (d.h. z.B. deutschen) Sätzen formulieren, welche Elemente man haben möchte.
4. Beginnen, diese Wünsche in XPath-Fragen zu übersetzen: Pfad, Bedingung, Funktion (vgl. Video!)
5. Setzen Sie das Resultat im XPath-Feld vom Oxygen-Editor ein. Funktioniert die Abfrage? Falls nein, lesen Sie die Fehlermeldung und überlegen, wie man sie richtigstellen könnte.

2.3 Einsetzen der Abfragen im XSLT-Template

1. Wenn die Abfrage funktioniert, überlegen Sie, wo im XSLT-Dokument diese hineingehört. Setzen Sie den XPath-Ausdruck dort ein, aber vergessen Sie nicht, dass sie jetzt entsprechend entweder absolute oder relative Pfade brauchen, damit das korrekt funktioniert. Auch verlangt Ihr XSLT-Dokument das Präfix einen `t:`-Namespace vor jedes (!) abzufragende Elemente (d.h. im Falle eines absoluten Pfades muss das an jedes Unterelement mit dran!).

2. Schreiben Sie zunächst Test-Meldungen ins Stylesheet, falls Sie nicht sicher sind (z.B. den Text 'TEST' hineinschrieben und dann im Resultat-Dokument schauen, ob der irgendwo auftaucht) und klicken nach jeder Hinzufügung auf das 'Run'-Symbol (nicht erst alles machen und dann feststellen, dass nichts geht - dann ist es oft sehr schwer, den Fehler wieder zu finden. Suchen, lokalisieren und beheben Sie Fehler nach jedem Arbeitsschritt).

3. Wie das Einstellen der Transformation funktioniert, entnehmen Sie bitte dem entsprechenden Video.

4. Wenn Ihre Abfragen im Template funktionieren, setzen Sie die HTML-Darstellung um.

5. *Bonus:* Wenn die Basics funktionieren, überlegen Sie sich Verschönerungen oder Zusatzfeatures. Fragen Sie gern bei der Umsetzung nach.

3 Lösungsideen

Lösungsideen finden Sie auf den Folien, wo einige ähnliche Abfragen draufstehen. Wenn Sie ganz verloren sind, versuchen Sie zuerstmal, diese Anfragen ungefähr zu verstehen (was könnte man damit abfragen wollen?), d.h. formulieren Sie die Abfragen in natursprachliche/deutsche Sätze um. Dann probieren Sie sie an ihrem Beispieldokument aus. Alles wird sicherlich nicht funktionieren, weil die Abfragen wohl zum Großteil nicht auf Ihre Datenstruktur passen werden.

`substring()` geht in dem Anzeige-Dings von Oxygen nicht, da dieses keine Stringoperationen durchführt. Wenn der Ausdruck stimmt, bekommt man keine Fehlermeldung, aber auch kein Ergbnis. In XSLT geht das dann (hoffentlich). Außerdem gehen die String-Operationen (siehe Cheatsheet Zeichenketten-Funktionen) immer nur auf genau eine Zeichenkette und nicht etwa eine Menge.

Eine Pfad-Abfrage gibt eine Menge zurück. Prädikate, also Bedingungen, sind in eckigen Klammern. Dort drin kann ich eine Funktion abfragen, die ein Ergebnis liefern (z.B.) ja/nein. Funktionen können auch um den ganzen Ausdruck herum, doch dann wird im Normalfall auch keine Knotenmenge ausgegeben. Daher immer erst in einem normalen Satz formulieren, in dem steht, was man eigentlich will. Das dann dahingehend zerlegen, was davon Pfad, was Bedingung und was Funktion sein könnte. Dann erst den Ausdruck schreiben.

Lokaler / globaler Scope: Ausprobieren den Mauszeiger in ein `<div>` zu stellen und dann nur `p` in die XPath-Abfrage einzugeben: Man findet Knoten im lokalen Kontext. `//` gibt immer vom globalen Kontext aus an.

`not(//persname[@ref="Mina"])` gibt 'false' zurück, weil er fragt: Gibt es im ganzen Dokument einen `persname[@ref="Mina"]`, die Antwort ist ja, sofern es eben so eine Referenz in Ihrem Dokument gibt. Daraufhin verneint `not()` alles, d.h. es stimmt nicht, dass es keinen `persname[@ref="Mina"]` im Dokument gibt.