Übung 3

1. XPath (5 Punkte)

Gib je einen XPath-Ausdruck an, welcher auf dem Eingabedokument sentences.xml

1. die Namen aller Autoren in den <author>-Elementen ausgibt,
2. die durchschnittliche Anzahl der Autoren pro <citation>-Element berechnet,
3. angibt, wieviele Sätze auf Seite 2 vorkommen (d.h. page="2"),
4. das durchschnittliche Erscheinungsjahr (<date>) aller <citation>-Elemente berechnet,
5. die Satzlänge (in Anzahl Zeichen) des 3. Satzes im Eingabedokument berechnet (unabhängig davon, in welchem <citation>-Element er steht,
6. das erste Wort (bis zum ersten Leerzeichen) im 2. <string>-Element ausgibt,
7. die Titel aller <citation>s aus den Jahren 1998 bis 2000 ausgibt,
8. ausgibt, ob der <title> der letzten <citation> – unabhängig von der Anzahl der <citation>-Elemente im Eingabedokument – das Wort "newspaper" enthält

Schreibe zum Testen (= Lösung dieser Aufgabe) ein kleines XSLT-Stylesheet xpathtest.xsl, das die XPath-Ausdrücke nacheinander auswertet und die Ergebnisse ausgibt. Tip: Wenn ein Ausdruck nicht tut, was er soll: Klammerung beachten/hinzufügen! Du kannst das Stylesheet z.B. mit jEdit (Installationsanleitung für XSLT-Plugin: JEdit-XSLT.pdf) oder über die Kommandozeile (Shell, "DOS-Box" oder Eclipse project) mit folgendem Befehl testen (xalanProcess.java ist der workaround für den JDK 1.5-"Bug"; vgl. Folie 50): `java xalanProcess -IN sample.xml -XSL xpathtest.xsl`. Für diese Aufgabe ist ausnahmsweise keine Java-Testklasse nötig.

2. Korpusarbeit mit XSLT (10 Punkte)

Es soll ein Stylesheet entwickelt werden, welches mehrere gegebene NLP-Annotationen kombiniert. Dieses Stylesheet soll dann aus einer Java-Klasse heraus auf den in der zip-Datei enthaltenen Annotationen gestartet werden (natürlich auf den ausgepackten XML-Dateien).

Teil 1: XSLT-Code (8 Punkte)
Gegeben ist eine primäre XML-Datei sentences.xml, welche die Eingabedatei für das zu entwickelnde Stylesheet bilden soll. Mit Hilfe von darin enthaltenen Informationen sollen sekundäre XML-Annotationen, die mit einem Part-of-Speech-Tagger erzeugt wurden, geöffnet und darin enthaltene Informationen über Verben (und nur über diese!) in die Ausgabe kopiert werden. Verben in den sekundären XML-Annotationen sind alle <w>-Elemente, die (mindestens) ein <p>-Kindelement mit einem pos-Attribut haben, dessen Wert mit "VB" anfängt.

XSLT-Rumpfcode für die Lösung ist in der Datei xslt/verbcontexts.xsl gegeben. Es ist lediglich der Code für `<template match="sentences">` zu füllen – **das sind nur ca. 15 Zeilen XSLT**, wenn die Ausgabeelemente (<sentence>, <text>, <verbs>) direkt und mit der kompakten {}-Syntax für XPath in Attributwerten angegeben werden. Achtung: Da ein Satz mehrere VB*-Tags enthalten kann, muss über die w-Elemente der Sekundärannotation in einer Schleife iteriert werden!

Der Zugriff auf sekundäre Annotationen kann mit Hilfe der XPath-Funktion `document()` erfolgen. Dazu den Dateinamen mit String-Funktionen zusammen bauen: Aus dem Wert des Attributes `no="N"` (Satznummer) wird der Dateinamensbestandteil `"-sN-"`, aus dem Wert des Attributes `page="P"` wird der Dateinamensbestandteil `"-pP-"`, also z.B. `C02-1004-s14-p1-TnT.xml` für das Beispiel unten.

Ferner ist die gegenüber dem Eingabedokument modifizierte Ausgabestruktur zu beachten und zu implementieren (s. Beispiel), d.h. anderes Root-Element, bibliografische Informationen wie <author>, <title>, <date> etc. entfallen, jedoch wird der citation-Text als Attributwert von <sentence> ausgegeben

```
<?xml version="1.0"?>
<!-- Beispiel zu Aufgabe 2; Anfang der XML-Eingabe-Datei sentences.xml -->
<citationList>
  <citation>
    <string>Brill and Resnik (1994)</string>
    <authors>
      <author>E Brill</author>
      <author>P Resnik</author>
    </authors>
    <title>A rule-based approach to prepositional phrase attachment disambiguation</title>
    <date>1994</date>
    <booktitle>In Proceedings of COLING</booktitle>
    <pages>1198--1204</pages>
    <publisher>ACL</publisher>
    <location>Kyoto</location>
    <sentence no="14" page="1">Supervised methods are as varied as the Back-off approach
by Collins and Brooks (1995) and the Transformation-based approach by Brill and Resnik
(1994).</sentence>
  </citation>
  <citation>
    <string>Collins and Brooks (1995)</string>
  ... </citation>
 ...
</citationList>
```

Beispiel für das Format der Ausgabe:

```
<?xml version="1.0"?>
<verbcontexts docid="C02-1004">
  <sentence no="14" page="1" citation="Brill and Resnik (1994)">
    <text>Supervised methods are as varied as the Back-off approach by Collins and Brooks
(1995) and the Transformation-based approach by Brill and Resnik (1994).</text>
    <verbs>
      <w str="are" cstart="19" cend="21">
        <p pos="VBP" p="1.000000e+00"/>
      </w>
      <w str="varied" cstart="26" cend="31">
        <p pos="VBN" p="9.193703e-01"/>
        <p pos="JJ" p="8.062967e-02"/>
      </w>
    </verbs>
  </sentence>
  <sentence no="14" page="1" citation="Collins and Brooks (1995)">
  ... </sentence>
...
</verbcontexts>
```

w-Elemente, unverändert aus sekundärer Annotation data/C02-1004-s14-p1-TnT.xml kopiert

Teil 2: Java-Code und Java-Testklasse für die Stylesheetausführung (2 Punkte)
Binde das Stylesheet aus Teil 1 dieser Aufgabe basierend auf dem in der Vorlesung angegebenen Transformer-Code in eine Java-Klasse ein – ggf. mit einem dummy-Stylesheet, falls der Code aus Teil 1 nicht funktioniert wie gewünscht. Der Dateinamenpräfix (im Beispiel C02-1004) für die sekundären Annotationen (welche durch den XSL Transformer, nicht im Java-Code geöffnet werden) ist dem XSLT-Stylesheet per Stylesheet-Parameter aus dem Java-Code zu übergeben. Wie das geht bitte ggf. nachlesen in Java-API-Dokumentation zu `javax.xml.transform.Transformer`.