Crawling und Tokenisierung

Laden Sie mit dem Befehl `wget -r -w 1 -A .html -R index.html www.tagesschau.de` Nachrichten-Seiten von der Adresse `www.tagesschau.de` herunter. Die heruntergeladenen Dateien werden in einer rekursiven Verzeichnisstruktur mit Wurzelverzeichnis `www.tagesschau.de` gespeichert.

Schreiben Sie dann ein Programm `extract.py`, welches das Verzeichnis `www.tagesschau.de` durchwandert und aus den heruntergeladenen HTML-Seiten die reinen Texte der Zeitungs-Artikel mit Überschriften, aber ohne Werbung, Navigationselemente etc. extrahiert. Sie können hier die Python-Bibliotheken *html.parser* (und/oder *re*) verwenden. Verzeichnisse können Sie mit der Funktion *os.walk* durchwandern. Dateien mit dem Namen `index.html` können Sie ignorieren.

Die Ausgabe jeder Webseite beginnt mit der Zeile "¿¿¿Dateiname¡¡¡", wobei Dateiname durch den Namen der Datei ersetzt wird. Überschriften sollten auf einer separaten Zeile stehen.

Aufruf: `python extract.py www.tagesschau.de > text.txt`

Schreiben Sie außerdem ein Tokenisierungs-Programm, welches den gesamten extrahierten Text in Sätze und Tokens (Wörter, Satzzeichen, Klammern etc.) zerlegt, und dann jeden Satz mit Leerzeichen zwischen den Tokens in einer separaten Zeile ausgibt. Die Zeilen mit den Dateinamen sollten Sie ausfiltern.

Aufruf: `python tokenizer.py abbreviations text.txt > text.tok`

Der Tokenisierer soll Abkürzungen korrekt behandeln. Eine Liste von deutschen Abkürzungen finden Sie hier:
`http://www.cis.uni-muenchen.de/~schmid/lehre/Experimente/data/abbreviations`

Bei Zahlen, die nur aus Ziffern bestehen und von einem Punkt gefolgt werden, sollte der Punkt nicht abgetrennt werden, da es sich meistens um Ordinalzahlen handelt.

Satzzeichen, Klammern und Anführungszeichen sollten nur am Anfang und Ende von Tokens abgetrennt werden. Tokens, die intern auch andere Zeichen als Buchstaben und Ziffern enthalten, wie bspw. Fließkomma-Zahlen (12,345), Internetadressen (tagesschau.de), Bindestrich-Wörter (Haber-Bosch-Verfahren) und anderes (10%-ige, Nutzer*innen) sollten nicht zerlegt werden. Schließende Klammern und Anführungszeichen nach einem Satzpunkt gehören noch zum Satz.

Gehen Sie so vor:

- Fügen Sie am Anfang und Ende des Textes ein Leerzeichen hinzu.
- Ersetzen Sie mit einer `re.sub`-Operation Folgen von Whitespace-Symbolen (Leerzeichen, Tabulatoren, Newlines, etc.) durch 1 Leerzeichen.
- Trennen Sie in einer Schleife mit dem Operator `re.sub` so lange Satzzeichen,

Klammern etc. am Anfang und Ende der Tokens durch Einfügen eines Leerzeichens ab, bis keine weiteren Abtrennungen mehr möglich sind.

- Fassen Sie mit einer weiteren `re.sub`-Operation Abkürzungen wieder mit ihrem Punkt zusammen. Der sub-Befehl matcht einen Punkt und das vorhergehende Token. Als zweites Argument übergeben Sie dem sub-Operator den Namen einer Funktion. Diese Funktion erhält das Match-Objekt als Argument, prüft ob eine Abkürzung gematcht wurde und gibt einen passenden Ersetzungs-String zurück.
- Zum Schluss fügen Sie mit einer weiteren sub-Operation Newlines als Satzgrenzen ein. Dabei müssen Sie berücksichtigen, dass noch (zuvor abgetrennte) Klammern und Anführungszeichen folgen können.
- Dann geben Sie den tokenisierten Textstring aus.

Versuchen Sie, kurzen und leicht verständlichen Code zu schreiben.

Sie dürfen Python-Bibliotheken, die nicht zu den Standard-Bibliotheken gehören, in dieser und anderen Übungen nur verwenden, wenn sie explizit erlaubt wurden. Außerdem müssen Sie die Programme selbstverständlich selbst schreiben, wobei Gruppenarbeit von bis zu 3 Personen erlaubt ist.

Schicken Sie die beiden Programme mit dem Betreff "EET: Abgabe 1" an `schmid@cis.lmu.de`.

Überlegen Sie sich außerdem schon einmal, welche Sprache und welche morphologischen Phänomene Sie in der nächsten Übungsaufgabe behandeln wollen.