Übung 04: Verarbeitungen grosser Datenmengen

Programmiertechniken in der Computerlinguistik II, FS 17

Abgabedatum: 25. April 2017, 23:59

Hinweise zur Abgabe

- Bitte gib jedes Python-Programm in einer eigenen Datei ab, die die Dateiendung `.py` hat und *ausführbaren* Python-Code enthält. Die Programme sollten importierbar sein.
- Gib all deine Antworten, welche keine Skripts sind, in einem PDF-Dokument oder einer Plain-Text-Datei (benannt nach dem Schema *vorname_nachname_uebung0X.pdf*) ab
- Geize nicht mit Kommentaren direkt im Programm-Code, wo Erläuterungen angebracht sind. Umfangreiche Erklärungen werden hingegen besser in einer separaten README-Datei mitgeliefert (vorzugsweise Plain-Text oder PDF).
- Halte dich an die Vorgaben des Python Style Guide! Grobe Verstösse werden mit Punkteabzug geahndet.
- Wie schon erwähnt, werden die Aufgaben diesmal per Git abgegeben! Auf OLAT musst du nur einen Link zu deinem Git-Repository hochladen!

Allgemeines zu dieser Übung

- Skalierbarkeit ist in dieser Übung ein wichtiges Kriterium, ein Skript, das nicht ordentlich mit grossen Daten umgehen kann, gilt als **nicht funktionierend.** Erinnere dich an die in der Vorlesung gezeigten Methoden.
- Plane dir genug Zeit für die Downloads der Korpora ein, es handelt sich hier um mehrere Gigabyte an Daten.
- Baue Abbruchbedingungen in dein Skript ein, so dass du nicht immer die Verarbeitung des gesamten Korpus abwarten musst um Ergebnisse zu sehen.
- Arbeite absturzsicher! Bei der Arbeit mit so grossen Daten kann es zu Abstürzen kommen, daher solltest du während dem Testen im Hintergrund besser keine grosse Arbeit mit ungespeicherten Änderungen offen haben.

1 Deduplizierung mit Zählen

In der ersten Aufgabe wirst du ein Skript schreiben, welches einen Korpus einliest und die 20 häufigsten Sätze in eine Textdatei schreibt.

1.1 Das Korpus

Für diese Aufgabe verwenden wie den Text+Berg-Korpus. Du kannst dir alle nötigen Dateien hier herunterladen (Grosser Download, nicht mobil herunterladen!). Entpacke den Unterordner 'SAC' (Unter 'Corpus_XML') in ein Verzeichnis deiner Wahl (Am Besten, wo du auch dein Skript hast). Mach dich mit der Struktur der Daten in den XML-Dateien vertraut.

1.2 Das Skript

Schreibe in deinem Skript eine Funktion `getfreqwords(indir, outfile)`, wobei `indir` ein Verzeichnis mit XML-Dateien ist und `outfile` eine Textdatei für die Ausgabe.
Dein Skript sollte durch alle XML-Dateien vom Schema 'SAC-Jahrbuch_XXXX_mul.xml' durchgehen und aus den Lemmata aller Token eines jeden Satzes einen 'lemmatisierten Satz' bilden. Das Skript soll herausfinden, welches die 20 häufigsten Sätze **mit wenigstens 6 Tokens** sind und diese in die Ausgabedatei schreiben.
Wichtig: Der Code muss skalierbar sein, auch Korpora die mehrmals so gross sind wie Text+Berg sollten so verarbeitet werden können! Skalierbarkeit kannst du überprüfen, indem du deine Systemwerte im Auge behältst (Bei Linux im System Monitor, bei Windows im Task Manager). Achte auf das RAM, sollte dieses während dem Laufen des Skripts dauernd ansteigen, wird es ab einer bestimmten Input-Grösse nicht mehr ausreichen (Und dein Skript automatisch abgebrochen...im besten Fall).

1.3 Tips

- Das Modul glob aus der Standard-Bibliothek von Python gibt dir die Möglichkeit, alle Dateien eines bestimmten Namenschemas in einem Verzeichnis als Liste zu erhalten.
- Nutze das Modul lxml um die XML-Bäume der Korpusdateien leicht durchsuchen zu können.

Abgabe: Ein Python-Skript mit der Funktion, wie immer als Modul importierbar.

2 Randomisierung

In dieser Aufgabe wirst du aus einem Korpus eine bestimmte Anzahl von zufälligen Elementen in eine erste Ausgabedatei schreiben, alle anderen in eine Zweite.

2.1 Das Korpus

Für diese Aufgabe verwenden wir einen Dump aller deutschen Wikipedia-Artikel. Du kannst ihn entweder selbst unter diesem Link herunterladen, oder dein Skript liest den Korpus direkt mit `urllib` bzw. `urlopen` aus dem Internet ab. Den Korpus kannst du direkt mit dem bz2-Modul öffnen und mit `lxml` auslesen (Es handelt sich dabei um eine einzelne riesige XML-Datei).

2.2 Das Skript

Schreibe nun eine Funktion `gettitles(infile, testfile, trainfile, k)` wobei `infile` das Korpus ist, der schon durch `bz2` geöffnet wurde, `testfile` die Ausgabedatei für `k` zufällig ausgesuchte Artikeltitel, und `trainfile` die Ausgabedatei für alle anderen Titel ist.
Diese Funktion soll das gegebene Korpus (eine einzelne XML-Datei) nach allen Artikeltiteln durchsuchen (gekennzeichnet mit dem Tag `{http://www.mediawiki.org/xml/export-0.10/}title`) und `k` Titel zufällig in die Testfile schreiben.
Wichtig: Der Code muss skalierbar sein, auch Korpora die mehrmals so gross sind wie der Wikipedia-Dump sollten so verarbeitet werden können! Skalierbarkeit kannst du überprüfen, indem du deine Systemwerte im Auge behältst (Bei Linux im System Monitor, bei Windows im Task Manager). Achte auf das RAM, sollte dieses während dem Laufen des Skripts dauernd ansteigen, wird es ab einer bestimmten Input-Grösse nicht mehr ausreichen (Und dein Skript automatisch abgebrochen...im besten Fall).

2.3 Tip

Verwende erneut `lxml` um das Korpus zu durchsuchen. Die Funktion `iterparse` hilft dir, den Baum zu durchsuchen.

Reflexion/Feedback

a) Fasse deine Erkenntnisse und Lernfortschritte in zwei Sätzen zusammen.

b) Wie viel Zeit hast du in diese Übungen investiert?