Übung 4: Dictionaries und reguläre Ausdrücke

Programmiertechniken in der Computerlinguistik I, HS 15

Abgabe: 11. November 2015 - 18:00

Hinweise zur Abgabe

- Bitte gib jedes Python-Programm in einer eigenen Datei ab, die die Dateiendung `.py` hat und *ausführbaren* Python-Code enthält.
- 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).
- Um das Hochladen der Abgabe auf OLAT zu erleichtern, kannst du die Dateien mit `zip` oder `tar` (oder einem anderen verbreiteten Format) archvieren / komprimieren.
- Jede Datei sollte den Namen sowie OLAT-Benutzernamen der Autoren enthalten.

1 Theoriefragen

Beantworte die folgenden Fragen zu den Themen der aktuellen Vorlesungen:

a) Für welche Aufgaben in der Computerlinguistik eignen sich Listen, Dictionaries und Tupel in Python ? Gib für jeden Typ 1-2 Beispiele an.

b) Welche Unterschiede gibt es in der Verwendung von regulären Ausdrücken in grep und in Python? Veranschauliche die Unterschiede an 2-3 Beispielen.

c) Welche Vor- und Nachteile haben verschiedene Codierungen, insbesondere UTF-8, Latin-1 und ASCII? Gebe je drei Vor- und Nachteile der drei Codierungen an, insbesondere im Bezug auf mehrsprachige Texte.

2 Stemming

Erstelle ein Programm, welches eine Textdatei mit je einem englischen Verb pro Zeile einliest und danach bei jedem Wort den Stamm und die Endung mithilfe von regulären Ausdrücken voneinander trennt.

Die Ausgabe erfolgt auf der Standardausgabe und sollte wie folgt aussehen:

```
testing  ->  test ing
listens ->  listen s
followed ->  follow ed
claim ->  claim
```

Erweitere das Programm, sodass es zusätzlich als Filter funktioniert, ähnlich wie die Werkzeuge aus Übung 1, zum Beispiel `grep`.

3 Identische Wörter erkennen

Erstelle ein Programm, welches, ähnlich wie in der Vorlesung vorgestellt, zwei Textdateien vergleicht und die Wörter ausgibt, welche in beiden Dateien vorkommen. Zusätzlich zu den gemeinsamen Wörtern sollen auch deren jeweilige Häufigkeiten ausgegeben werden.

Die Wörter und die Häufigkeiten sollten in zwei Varianten welche als Argumente von der Kommandozeile mitgegeben wurden, auf der Standardausgabe ausgegeben werden:

- Sortiert nach Alphabet
- Sortiert nach Häufigkeit

4 Wortarten erkennen und annotieren

Diese Übung basiert auf der Aufgabe *Wortendungen erkennen* aus Übung 2.

Das Programm sollte nun so erweitert werden, dass erkannte Wörter in einer Datei zwischengespeichert werden, währenddessen die unbekannten Wörter mithilfe von regulären Ausdrücken erraten und danach vom Nutzer bestätigt oder korrigiert werden können.

Wie können Wörter gespeichert werden, welche eine Wortform haben, aber zu mehreren Wortarten gehören können?

Als Eingabe sollen zwei Möglichkeiten angeboten werden:

- Eingabe eines Satzes direkt von der Kommandozeile als Argument
- Eingabe von Sätzen aus einer Textdatei, welche als Argument gegeben wird.

Mögliche Syntax für die beiden Aufrufe:

- `python aufgabe_3.py --database tagged.txt --text 'Dies ist ein Test.'`
- `python aufgabe_3.py --database tagged.txt --file input.txt`

Die Ausgabe der vollständig annotierten Sätze erfolgt auf der Standardausgabe, oder (optional) in eine weitere Textdatei. In beiden Fällen sollten die Resultate in einer geeigneten Form zwischengespeichert werden.

Als Tagset dient das STTS Tagset, in der Datei `tagged.txt` befinden sich einige Beispiele.

5 Zusatzaufgabe

Erweitere das Programm aus Aufgabe 4, sodass aus der Nutzereingabe mithilfe von regulären Ausdrücken auch noch weitere Formen, wie zum Beispiel Pluralformen oder flektierte Verben abgespeichert werden.

Welche Probleme ergeben sich aus dieser Erweiterung?

Reflexion/Feedback

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

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