Schlussprüfung Programmiertechniken in der Computerlinguistik I HS 13

Aufgabenstellung: Simon Clematide/Martin Volk

Prüfung vom 9. Januar 2014
Institut für Computerlinguistik, Universität Zürich

Vorname _____ Matrikelnummer _____

Nachname _____

Für Studierende der folgenden Studiengänge:

☐ BA - Studiengang Computerlinguistik (Phil. Fakultät)

☐ BA - Studiengang Computerlinguistik und Sprachtechnologie (Phil. Fakultät)

☐ Andere:

Nur für Lizenziatsstudierende der Computerlinguistik als ein Fach aus der Phil. Fakultät:

Strasse: _____ Hauptfach: _____

PLZ/Ort: _____ E-Mail: _____

Die Prüfungsresultate von ECL und PCL von den Lizentiatsstudierenden werden (zusammen) per Post verschickt. *Bitte Adresse nicht vergessen!*

Aufgabe Nr.:	1	2	3	4	5	6	7	8	9	10	11	12	**Summe**
Punktzahl:	12	10	4	4	14	4	14	5	2	4	2	15	90
Davon erreicht:													

Note SU: _____ Note SP: _____

Endnote: _____ Bestanden: ☐ Ja ☐ Nein

Auf jedes separate Blatt mit Lösungen den Nachnamen schreiben! Viel Erfolg!

Wichtige Hinweise

Maximale Punktzahl: 90 Pro Punkt sollte ungefähr 1 Minute gebraucht werden.
Hinweise: Bitte schreiben Sie in einem knappen, aber verbalen Stil (keine Stichwortsammlungen). Bei inhaltlichen Auswahlsendungen, wo einfach alles spontan hingeschrieben und Falsches wie Korrektes munter vermischt wird, behalten wir uns Abzüge vor. Erlaubtes Hilfsmittel ist die Referenzkarte zu Python. Die Antworten können **deutsch** (bevorzugt) oder **englisch** sein.

1. UNIX-Kommandos für Schlagzeilenkorpus (12 Punkte)

In einem Unterverzeichnis namens `2013/` befinden sich genau 365 Textdateien mit **der** Haupt-Schlagzeile einer Boulevardzeitung von jedem Tag.

Jede Datei ist nach folgendem Muster benannt: `MM-DD.txt` (MM=Monat numerisch, DD=Tag numerisch). Z.B. enthält `2013/06-01.txt` die Schlagzeile vom 1. Juni 2013.

Jede Datei hat genau eine Zeile, welche den **tokenisierten** Text der Schlagzeile enthält. Fiktives Beispiel:

```
SP-Politikerin heiratet SVP-Kantonsrat .
```

3 (a) Mit welchem Terminalbefehl (oder Befehlen) lassen sich alle Schlagzeilen des Jahres aneinandergehängt in eine Datei `2013.txt` ablegen?

```
$ cat 2013/*.txt > 2013.txt
```

2 (b) Mit welchem Terminalbefehl (oder Befehlen) lassen sich alle Schlagzeilen vom Monat Oktober aneinandergehängt auf dem Terminal anzeigen?

```
$ cat news2013/10*.txt
```

3 (c) Wie lautet der Grep-Befehl, um aus der Datei `2013.txt` alle **Bindestrich-Komposita** auf dem Terminal anzuzeigen, welche mit einem Parteikürzel ("SP", "CVP", "SVP", "FDP") beginnen. Z.B. soll "SP-Politikerin" oder "FDP-Bundesrat" einzeln auf einer Ausgabezeile erscheinen.

```
$ grep -Po '\b(SP|CVP|SVP|FDP)-\S+'
```

3 (d) Der Befehl `tr ' ' '\n'` kann verwendet werden, um alle Leerzeichen in Newlines zu verwandeln. Beschreiben Sie knapp und exakt, was der folgende Befehl auf dem Terminal **ausgibt** (also nicht, was die einzelnen Werkzeuge machen). Erfinden Sie ein Beispiel, wie die **erste** und **letzte** Zeile der Ausgabe aussehen könnte.

```
$ tr ' ' '\n' < 2013.txt | sort | uniq -c | sort -n
```

- Alle Types der Tokens aus 2013.txt (mit ihrer Vorkommenshäufigkeit)
- numerisch aufsteigend (!) geordnet.
- Bsp. 1. Zeile: `1 SVP-Kantonsrätin` ; letzte Zeile: `999 der`

1 (e) Wieso kann es Sinn machen, solche Aufgaben mit UNIX-Kommandos zu lösen und z.B. nicht in Python?

Weil es auf der Kommandozeile viel einfacher geht, solche einfachen Probleme zu lösen.

2. Python-Theorie (10 Punkte) Kreuzen Sie **alle** korrekten Aussagen an. Es gibt 0.5 Punkte pro richtige Antwort, 0.5 Punkte **Abzug** pro **falsche** Antwort.

2 (a) Objekte

- ■ **Jedes Attribut ist ein Objekt.**
- ☐ Jedes Objekt ist ein Attribut.
- ■ **Jede Methode ist ein Attribut.**
- ☐ Jedes Objekt ist eine Methode.

2 (b) Oberklassen

- ■ **Jede selbstdefinierte Klasse hat eine Oberklasse.**
- ☐ Unterklassen eines Objekts heissen Instanzen.
- ■ **Jedes Objekt ist eine Instanz einer Klasse.**
- ■ **Die oberste Klasse heisst `object`.**

2 (c) Zeichenketten

- ■ **Der String `"\n"` und der String `'\n'` sind die gleiche Zeichenkette.**
- ☐ Der String `"abc\n"` und `r"abc\n"` sind die gleiche Zeichenkette.
- ☐ Die Länge der Zeichenkette `"""das\tART\nHaus\tNN\n"""` ist grösser als 16.
- ■ **Solange in einem Stringliteral *s* nur ASCII-Zeichen vorkommen, ist es egal, ob wir das Stringliteral als `u's'` oder `'s'` schreiben.**

2 (d) Veränderliche Datenstrukturen

- ☐ Normale Strings (d.h. Objekte vom Typ `str`) haben im Gegensatz zu Unicode-Zeichenketten (d.h. Objekte vom Typ `unicode`) den Vorteil, dass sie veränderlich (*mutable*) sind.
- ■ **Die Schlüssel in Dictionaries (`dict`) müssen unveränderlich (*immutable*) sein.**
- ■ **Mengen (`set`) können nur unveränderliche Datenstrukturen als Elemente haben.**
- ☐ Die Werte in Dictionaries (`dict`) müssen immer unveränderlich sein.

1 (e) Identität

- ☐ Zwei Objekte `o1` und `o2`, welche identisch sind (d.h. es gilt `id(o1) == id(o2)`), haben immer denselben Wert (d.h. es gilt `o1 == o2`).
- ■ **Die beiden Variablen a und b sind identisch nach der Verarbeitung dieses Kodes.**

```
a = "a"
b = a
```

1 (f) Generatorausdrücke und Listenkomprehensionsausdrücke

- ☐ Die Funktion `len` lässt sich auf Generatorausdrücke anwenden.
- ■ **Die Funktion `len` lässt sich auf Listenkomprehensionsausdrücke anwenden.**

4 **3. Import-Statement (4 Punkte)** Was unterscheidet die beiden Versionen des import-Statements:

```
import nltk                 ### (A)
from nltk import *          ### (B)
```

A Ermöglicht es, alle Module, Packages, Objekte und Klassen aus nltk mit der Punktnotation nltk.XYZ zu benutzen.

B Ermöglicht es, dieselben Dinge wie in A zu benutzen (XYZ, ohne Präfix nltk.), d.h. sie werden direkt im globalen Namensraum zugänglich.

4 **4. Ausdrücke (*expressions*) vs. Anweisungen (*statements*) (4 Punkte)** Was ist der Unterschied und das Verhältnis zwischen Ausdrücken und Anweisungen?

- Ausdrücke evaluieren zu Objekten (falls nicht eine Ausnahme die Berechnung unterbricht) und können keine Anweisungen enthalten.
- Anweisungen werden ausgeführt, aber evaluieren nicht zu einem Wert. Anweisungen können Ausdrücke enthalten.

14 **5. Schlaufen (14 Punkte)** Ein Hacker hat Teile des Programms zur Formatierung von textuellen Häufigkeitstabellen böswillig mit Kommentaren der Form `#X#` überschrieben. Mit Hilfe eines Beispieloutputs sollte es Ihnen möglich sein, die Kommentare mit Variablen oder Literalen zu ersetzen, damit das Programm wieder lauffähig wird.

```
import nltk

cfd = nltk.ConditionalFreqDist(
   (genre, word)
   for genre in nltk.corpus.brown.categories()
   for word in nltk.corpus.brown.words(categories=genre))

genres = ['news', 'religion', 'hobbies']
modals = ['can', 'could', 'may', 'might', 'must', 'will']

def tabulate(cfdist, words, categories):
   print '%-16s' % #A#,
   for #B# in #C#:
      print '%6s' % #B#,
   print
   for #D# in #E#:
      print '%-16s' % #D#,
      for #F# in #G#:
         print '%6d' % cfdist[#H#][#I#],
      print

tabulate(cfd,modals,genres)
```

Output des Programms:

```
Category           can  could    may  might   must   will
news                93     86     66     38     50    389
religion            82     59     78     12     54     71
hobbies            268     58    131     22     83    264
```

`#A#`	`'Category'` (1Pkt.)	`#F#`	beliebige Variable wie `w` (1Pkt)
`#B#`	beliebige Variable wie `w` (1Pkt)	`#G#`	`categories` (2Pkt)
`#C#`	`words`(2Pkt)	`#H#`	wie `#D#` z.B. `c` (2Pkt)
`#D#`	beliebige Variable wie `c` (1Pkt)	`#I#`	wie `#F#` z.B. `w` (2Pkt)
`#E#`	`categories` (2Pkt)		

4 **6. Ausgabe von Ausdrücken mit Listen und anderen Sequenzen (4 Punkte)**

```
>>> names = ['Carla', 'Bruno', 'Adam']
>>> print names[-1][0]          Ausgabe:  A
>>> names.sort()
>>> print names[0]              Ausgabe:  Adam
>>> del names[0]
>>> print names[0]              Ausgabe:  Bruno
>>> names.append('Daniel')
>>> print names                 Ausgabe:  ['Bruno', 'Carla', 'Daniel']
```

7. Trunkierung von Wörtern (14 Punkte) Trunkierung ist eine alte Technik im *Information Retrieval* (IR), um die Anzahl der Indexterme eines Dokuments klein zu halten. Bei der Trunkierung auf eine feste Anzahl Zeichen werden alle Tokens von ihrem Ende her auf eine maximale Anzahl Buchstaben gekürzt, z.B. auf 6 Buchstaben. Weiter sollen **einbuchstabige** Tokens wie im IR üblich ignoriert werden. Folgender Beispielkode illustriert dies:

```
>>> doc = ['Alle','Dokumente','dieser','Dokumentensammlung',':']
>>> trunk(doc,6)
['Alle','Dokume','dieser','Dokume']
```

Die Funktion `trunk(l,n)` evaluiert zur selben Liste wie `l`, wobei alle Strings auf eine maximale Länge `n` gekürzt sind. Hinweis: `"b"[:4]` evaluiert zu `"b"` und `"abc"[:2]` zu `"ab"`.

Definieren Sie nun die **Funktion** `trunk`, zuerst ohne Listenkomprehension, dann mit.

7 (a) Ohne Listenkomprehension

```
def trunk1(l,n):
    """ Trunkierung ohne Listenkomprehension """
    result = []
    for t in l:
        if len(t) > n:
            t = t[:n]
        if len(t) > 1:
            result.append(t)
    return result
```

7 (b) Mit Listenkomprehension

```
def trunk2(l,n):
    """ Trunkierung mit Listenkomprehension

        Hinweis: "b"[:4] ==> "b"
    """
    return [t[:n] for t in l if len(t) > 1]
```

5 **8. Reguläre Ersetzung (5 Punkte)** Gegeben sei eine Sammlung von HTML-Dateien mit der folgenden Struktur:

```
...
<div class="content">Bla ... bla</div>
<div class="footer">2014. All rights reserved. </div>
...
```

Definieren Sie einen regulären Such- und Ersetzungsausdruck, der einen String (`htmlstring`) mit dem Inhalt einer ganzen HTML-Datei reduziert auf den textuellen Inhalt des HTML-Tags `<div class="content">`, d.h. auf *Bla ... bla*.

```
pattern = _____________________________________________________________

replacement = _________________________________________________________
text = re.sub(pattern,replacement,htmlstring)
```

```
pattern = r'^.*<div class="content">(.*?)</div>.*$'

replacement = r'\1'
text = re.sub(pattern,replacement,htmlstring)
```

2 **9. Wozu ist das Regex-Flag `(?u)` nützlich in der Sprachverarbeitung? (Antwort mit einem Beispiel) (2 Punkte)**

Es macht, dass das Metazeichen `\w` alle alphanumerischen Zeichen von Unicode matcht und nicht bloss die ASCII-Zeichen. Z.B. Umlaute im Deutschen.

4 **10. Pipeline für Webseiten-Verarbeitung (4 Punkte)** Welche Schritte der Verarbeitung sind typischerweise notwendig, um eine einfache korpuslinguistische Verarbeitung einer HTML-Datei hinter einer Web-Adresse durchführen zu können (z.B. Vokabulargrösse bestimmen)?

- Download der Datei
- Trimmen auf gewünschten Inhalt
- Verwandlung von HTML in Text
- Korpuslinguistische Aufarbeitung (Tokenisierung usw.)

2 **11. Was ist der Unterschied zwischen lokalen und globalen Namen? (2 Punkte)**

- Globale Namen sind innerhalb eines Moduls sichtbar, sobald sie definiert sind.
- Lokale Namen sind innerhalb einer Funktion oder Methode sichtbar, sobald sie definiert sind (auch via Parameterliste definierbar).

15 12. Buchstabenstatistik (15 Punkte)

Definieren Sie eine Funktion `tabulate_chars`, die eine Zeichenkette einliest und dann eine textuelle Häufigkeitstabelle aller Buchstaben ausgibt. Die Tabelle soll absteigend nach Häufigkeit geordnet sein. Die Reihenfolge bei gleichhäufigen Buchstaben ist nicht spezifiziert, d.h. frei. **Leerzeichen** müssen ignoriert werden.

```
>>> tabulate_chars("ottos mops trotzt")
t 5
o 4
s 2
m 1
p 1
r 1
z 1
```

```
def tabulate_chars(s):
   chars = {}
   for c in s:
      if c in chars:
         chars[c] += 1
      else:
         chars[c] = 1
   del chars[' ']
   for c in sorted(chars, key=chars.get, reverse=True):
      print c, chars[c]
```