

Prüfung Programmiertechniken in der Computerlinguistik I HS 2014

Aufgabenstellung: Simon Clematide/Martin Volk

Prüfung vom 8. Januar 2015
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)
☐ Anderer Studiengang: _____

Aufgabe Nr.:	1	2	3	4	5	6	7	8	9	10	Summe
Punktzahl:	18	15	12	4	5	7	6	8	5	10	90
Davon erreicht:											

Note SU: _____ Note SP: _____

Endnote: _____ Bestanden: ☐ Ja ☐ Nein

Wichtige Hinweise

- Maximale Punktzahl: 90
- Pro Punkt sollte ungefähr 1 Minute gebraucht werden.
- Bitte schreiben Sie in einem knappen, aber verbalen Stil (keine Stichwortsammlungen). Bei inhaltlichen Auswahlendungen, 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. Bei sprachlichen Verständnisfragen melden Sie sich bei der Aufsichtsperson.

Auf jedes separate Blatt mit Lösungen den Nachnamen schreiben!

Viel Erfolg!

1. Auf der Kommandozeile mit dem Text+Berg-Korpus arbeiten (18 Punkte)

Gegeben seien 48 Jahrbücher des Text+Berg-Korpus von 1900 bis 1949 wie in der Übung 0 von PCL I. Es gibt Tokenzeilen und Strukturzeilen. Tokenzeilen enthalten in 3 tabulator-separierten Spalten das Token, das POS-Tag und das Lemma. Strukturzeilen markieren den **Anfang** von Zeitschriftenartikeln und Sätzen und beginnen mit "<article " bzw. "<s ".

```
<article n="1">
<s lang="de" n="1-1">
Vorwort      NN      Vorwort
<s lang="de" n="1-2">
Das          ART      d
Jahrbuch     NN      Jahrbuch
XXXVI       CARD     @card@
weicht      VVFIN     ab+weichen
ab          PTKVZ     ab
.           $.      .
<s lang="de" n="1-3">
Dementsprechend ADJD   dementsprechend
```

Alle Dateien sind gemäss Beispiel "SAC-Jahrbuch_1900_mul_column.txt" benannt. Die folgenden Fragen gehen davon aus, dass die Befehle im Verzeichnis abgesetzt werden, wo die Jahrbücher liegen. Bei grep können Sie die Option "-E" oder "-P" hinschreiben, um anzudeuten, dass Sie die erweiterten regulären Ausdrücke verwenden, welche Sie auch von Python kennen.

- (a) Wie bestimmt man im Terminal mit Hilfe von UNIX-Kommandos die Anzahl Zeitschriftenartikel im Jahrbuch von 1900? 2

.....

- (b) Wie bestimmt man im Terminal die Gesamtanzahl aller Zeitschriftenartikel in allen Jahrbüchern? 3

.....

- (c) Wie kann man jedes (deutsche) Partizip Perfekt (VVPP POS-Tag) im Terminal ausgeben, das mit ,ge' beginnt und auf ,t' endet, wobei die Grossschreibung keine Rolle spielen soll? 4

.....

- (d) Wie kann man im Terminal eine Häufigkeitsstatistik aller POS-Tags erstellen? D.h. einen Output ausgeben, wo für jedes POS-Tag ersichtlich ist, wie oft es insgesamt in allen Jahrbüchern zusammen vorkommt? 4

.....

.....

.....

- (e) Wie kann man im Terminal alle Zeilen ausgeben, bei denen sich die Wortform nur durch eine s-Endung oder es-Endung vom Lemma unterscheidet? 5

.....

.....

.....

2. Python-Datenstruktur für das Text+Berg-Korpus (15 Punkte)

Wie könnte eine geeignete und einfache Repräsentation (=Datenstruktur) in Python aussehen für die Korpusdaten eines Jahrbuchs von Aufgabe 1 dieser Prüfung? Folgende Bedingungen müssen erfüllt sein:

- Die Artikelstruktur wird ignoriert.
- Für jedes Token muss das POS-Tag und das Lemma abgespeichert sein.
- Für jeden Satz muss die Sprache bestimmbar sein.
- Man soll mit normaler Python-Indexierung, d.h. mit der `[]`-Notation auf die Elemente zugreifen können, z.B. die Wortart vom letzten Token in einem Satz berechnen.

- (a) Beschreiben Sie in natürlicher Sprache, wie Sie die Korpusdaten in Python repräsentieren. Begründen Sie kurz und erwähnen Sie allfällige Vor- bzw. Nachteile.

5

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

- (b) Schreiben Sie einen Python-Ausdruck hin, der die Beispielzeilen von Aufgabe 1 in der von Ihnen gewählten Datenstruktur repräsentiert.

5

.....

.....

.....

.....

.....

.....

- (c) Angenommen das Jahrbuch 1900 wäre in Ihrem Format in der Variable `corpus1900` abgelegt. Schreiben Sie den Python-Kode auf, um herauszufinden, wie viele deutsche Sätze im Jahrbuch weniger als 3 Token haben.

5

.....

.....

.....

.....

.....

.....

3. Python-Theorie (12 Punkte)

Kreuzen Sie **alle** korrekten Aussagen an. Jede richtige Antwort ergibt 0.5 Punkte, jede **falsche** Antwort ergibt 0.5 Punkte **Abzug**. Man kann minimal 0 Punkte machen pro Teilaufgabe. Es können beliebig viele Antworten richtig oder falsch sein.

(a) Funktionen

2

- ☐ Funktionen müssen nicht zwingend ein return-Statement enthalten.
- ☐ Wenn im Rumpf einer Funktionsdefinition zuerst ein Stringliteral steht, wird dies als Online-Hilfe der Funktion verwendet.
- ☐ Wenn eine Funktion mit 2 Argumenten definiert wurde, kann man sie auch nur mit 1 Argument aufrufen. Das 2. Argument wird dann implizit als None gesetzt.
- ☐ Die Namensräume für Funktionen und andere Objekte sind unterschiedlich. Folgender Code gibt "Test" aus:

```
def foo():
    print "Test"
foo = 1
foo()
```

(b) Klassen

2

- ☐ Methoden sind auf der Ebene von Klassen definiert.
- ☐ Vererbung bezweckt, dass nicht jede Klasse alle ihre Methoden definieren muss.
- ☐ Alle Objekte sind Instanzen einer Klasse.
- ☐ Die grundlegende Klasse in Python heisst `object`.

(c) Objekte

2

- ☐ Jedes Attribut ist ein Objekt.
- ☐ Attribute werden standardmässig in den Instanzen gespeichert.
- ☐ Listen sind in Python keine Objekte, sondern spezielle Datenstrukturen.
- ☐ Dictionaries sind in Python keine Objekte, sondern spezielle Datenstrukturen.

(d) Zeichenketten

2

- ☐ Der String `r"\t"` hat eine Länge von 2 Buchstaben.
- ☐ Zeichenketten, die mit einfachem Apostroph `'` begrenzt sind, dürfen sich über mehrere Zeilen im Quellcode erstrecken.
- ☐ UNICODE-Escape-Sequenzen der Form `\uNNNN` werden in einer Zeichenkette wie `ur'Ein\u0020A'` zu einem Buchstaben (hier Leerzeichen) aufgelöst.
- ☐ Eine Zeichenkette `s`, welche UTF-8-kodierte Bytes enthält, kann mittels `s.decode('utf-8')` in den entsprechenden `unicode`-String verwandelt werden.

(e) Veränderliche Datenstrukturen

2

- ☐ Zeichenketten vom Typ `str` sind in Python veränderlich.
- ☐ Ein Python-Ausdruck für ein Tupel wie `('der', [103, 1014])` darf als Schlüssel in einem Dictionary verwendet werden.
- ☐ Mengen (`set`) können nur unveränderliche Datenstrukturen als Elemente haben.
- ☐ Tupel dürfen keine Listen als Element enthalten.

(f) Identität und Wertgleichheit

1

- ☐ Wenn 2 Objekte denselben Wert haben (`==`), dann sind sie auch identisch (`is`).
- ☐ Wenn 2 Objekte identisch (`is`) sind, dann haben sie auch denselben Wert (`==`).

(g) Listenkomprehensionsausdrücke

1

- ☐ Auf Listenausdrücke lassen sich Methoden anwenden mit der Punktnotation, z.B. so:

```
[x.lower() for x in "ABC"].index("b")
```
- ☐ Listenkomprehensionsausdrücke können im Bedingungssteil Print-Anweisungen enthalten.

4. Erklären Sie den Unterschied von Ausdrücken und Anweisungen in Python (4 Punkte)

4

.....

.....

.....

.....

.....

.....

.....

5. Lokale Namensräume (5 Punkte) Was versteht man in Python unter lokalen Namensräumen? Geben Sie 2 unterschiedliche Beispiele, wie lokale Namen entstehen können.

5

.....

.....

.....

.....

.....

.....

.....

.....

6. Reguläre Ausdrücke in Python (7 Punkte) Im Gutenberg-Projekt werden in der Textversion von Büchern manchmal Notationen im Markdown-Stil verwendet. Wie in der Tabelle unten ersichtlich wird mit Tilde *kursiver Text* und mit Dollar **Fettdruck** markiert.¹

7

Markdown-Notation	Notation mit HTML-Tags (=markup)
Kursiver Text ist ~als kursiv~ und Fettdruck \$als fett\$ markiert.	Kursiver Text ist <i>als kursiv</i> und Fettdruck als fett markiert.

Schreiben Sie in Python eine Funktion `markdown2html(string)`, welche die Markdown-Notation (wie links) in eine Zeichenkette mit HTML-Tags (wie rechts) überführt und als Funktionswert zurück gibt.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

¹<http://www.gutenberg.org/files/42489/42489-0.txt>

7. Manipulation von Listen (6 Punkte)

6

Das Folgende sind Eingaben im Python-Interpreter, welche nacheinander ausgeführt werden. Schreiben Sie die Ausgabe der Print-Statements auf die entsprechenden Linien.

```
>>> names = ['Xenia', 'Carla', 'Rosa']
>>> print [n[0] for n in names]      Ausgabe: _____
>>> names2 = names
>>> names.sort()
>>> print names[1]                  Ausgabe: _____

>>> print names2[1]                  Ausgabe: _____
>>> names = ['Xenia', 'Carla', 'Rosa']
>>> print names.append('Daniela')    Ausgabe: _____

>>> print names[-1]                  Ausgabe: _____
>>> names2 = names
>>> del names[0]
>>> print names2 is names            Ausgabe: _____
```

8. Übersetzen (8 Punkte)

8

Ein Hacker hat Teile eines einfachen Übersetzungsprogramms böswillig gelöscht. Mit Hilfe des folgenden Beispielsoutputs sollte es Ihnen möglich sein, das Programm wieder lauffähig zu machen.

```
Input: Tante Polly sah den Mann mit dem Fernglas
Translation: aunt <<<Polly>>> saw the man with the telescope
```

Schreiben Sie den fehlenden Code direkt auf die leeren Linien. Für die Übersichtlichkeit ist das Lexikon alphabetisch geordnet auf den Keys.

```
my_lexicon = {
    'Fernglas': 'telescope',
    'Mann': 'man',
    'Tante': 'aunt',
    'dem': 'the',
    'den': 'the',
    'der': 'the',
    'im': 'in',
    'mit': 'with',
    'sah': 'saw'}

test_sentence = 'Tante Polly sah den Mann mit dem Fernglas'
print 'Input:', test_sentence

test_list = _____

print 'Translation:',

for word in test_list:
    _____

    print _____
else:
    print _____
```

9. Init-Methode (5 Punkte) Wozu dient die `__init__()`-Methode bei selbstdefinierten Klassen? Wann wird sie aufgerufen?

5

[illegible]

10. Kontextabhängig Übersetzen (10 Punkte)

10

Die Firma "troogle" bietet ein gleichnamiges Python-Package an, mit dem man kontextabhängig die wahrscheinlichste deutsche Übersetzung für ein einzelnes englisches Wort nachschlagen kann. Die Funktion `lookup(word, preceding, succeeding, translated_preceding)` liefert die beste deutsche Übersetzung für Wort `word`, wobei gewisse Kontextinformation berücksichtigt wird. Je **mehr Information** mitgegeben wird, umso **besser** wird die Übersetzung.

- Das Argument `preceding` ist das vorangehende englische Token oder `None`, falls es keines gibt.
- Das Argument `succeeding` ist das nachfolgende englische Token oder `None`, falls es keines gibt.
- Falls die deutsche Übersetzung des vorangegangenen Worts bereits bekannt ist, wird es als Argument `translated_preceding` mitgegeben, ansonsten `None`.

Der Beispielaufruf, um die Übersetzung von “the” in einem Satz wie “the man with a telescope...” nachzuschlagen lautet: `lookup('the', None, 'man', None)` und würde idealerweise zu “der” evaluieren. Im Satz “the woman with a telescope” würde man die Zeichenkette “die” als Resultat des Lookups erwarten.

Definieren Sie eine Funktion `lookup_sentence(sent)`, welche einen englischen Satz (=Liste von Tokens) von links nach rechts mit Hilfe von `lookup` möglichst gut in einen deutschen Satz (=Liste von Tokens) übersetzt und ihn als Funktionswert zurückliefert. Bitte auch Sätze, die nur aus 1 Wort bestehen, korrekt behandeln. Achten Sie auf die korrekte Behandlung von Wörtern am Anfang und Ende des Satzes.

[illegible]