PCL1-Loesungen-Uebung4 LennartVonThiessen,lvthiessen, RolandBenz,rolben,97923163

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.

Dictionaries: In dieser Datenstruktur werden die Keys indexiert (dh. der gesuchte Key-String ist Input in die Hashfunktion, Output der Hashfunktion ist der Index des Arrays im Hintergrund ...), somit ist der Zugriff auf ein Key-Value Paar sehr schnell möglich, da im Idealfall keine eigentliche Suche nötig ist, sondern bloss eine einfache Moduloberechnung, um den gesuchten Speicherinhalt zu finden. **In CL einsetzbar z.B. für die Bearbeitung riesiger Wortlisten; oder Lexiken auf die immer wieder nach dem Key gesucht wird. Im Value des Key-Value Paars könnte je nach Einsatz ein konkatenierter String oder ein Tupel gespeichert werden.**
Tupel: Tupel, werden in der Geometrie Vektor genannt. Die Anzahl Dimensionen der Tupel sind bei der Deklaration festzulegen oder werden durch die Initialisierung implizit festgelegt und können danach nicht mehr verändert werden. Dadurch eignen sich Tupel zum Festlegen eines Modells in mehreren Dimensionen. **In CL einsetzbar z.B. für 3D-Tupel (Wort, X, Y) mit X,Y z.B.: POS-Tag, oder Morphologische Kategorie, oder Syntaktische Funktion, oder Sprache, oder Suffix, oder Lemma usw.**
Listen: Listen sind intern als eine Aneinanderreihung von Graphen umgesetzt, wo jeder Knoten die Adresse seines Nachfolgers und ev. des Vorgängers gespeichert hat. Der Vorteil dieser Datenstruktur ist, dass beliebig neue Knoten an beliebigen Stellen eingefügt werden können. Der Nachteil ist, dass diese Knoten (bzw. die gespeicherten Daten-Objekte dahinter) im Speicher verstreut sind und sich somit Suche und Sortierung verlangsamen. **In CL einsetzbar z.B. als kleine Objektstruktur in einem Directory, oder um neu erzeugte Subdirectories nach einer Suche im Gesamtdirectory zu sortieren, oder ganz allgemein wenn die Datenmenge klein ist, oder die Zeit vernachlässigbar ist, oder die Flexibilität und Reihenfolge höher gewichtet wird als der Geschwindigkeitsvorteil beim Directory.**

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

Es gibt verschiedene Optionen für grep wie reguläre Ausdrücke interpretiert werden, wie z.B. grep -p. Diese Option (-p) interpretiert die Regex als Perl Ausdrücke.

Grundsätzlich kann man sagen, dass die Python Bibliothek re viel benutzerfreundlicher ist, als grep. Die Bibliothek hat verschiedene Methoden (match, search, findall usw.) mit ihren jeweiligen Argumenten, wobei einige Argumente eine Flagfunktion haben, dh. Sie haben eine ähnliche Funktion wie die Optionen (-i, -o usw) bei grep.

Zudem bietet die re-Bibliothek allgemein mehr Möglichkeiten. Beispielsweise kann man mit der Option re.U in der Re.search() Methode festlegen, dass das Regex-Muster „\w“ auch äöü matcht. Bei grep muss man für diesen Vorgang eine Klasse bilden [\wäöü]. Noch interessanter ist die Suche über mehrere Zeilen mit der Option re.MULTILINE. Der folgende Ausdruck sucht in den drei Zeilen des Inputstrings nach einem X am Anfang und erzeugt einen Match: var=re.search('^X', 'A\nB\nX', re.MULTILINE) # Match. Mit dem einfachen Einzeilen-grep ist so etwas nicht möglich.

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.

Hintergrund:

Man verwendet in diesem Zusammenhang die Begriffe Zeichensatz (character sets), Windows Zeichensatztabelle (codepage) und Kodierungssysteme (character encoding systems). **Latin-1** (ISO/IEC 8859-1) ist ein Standard zur 8 bit Zeichenkodierung. Der Zeichensatz entspricht der Windows Zeichsatztabelle 850 und enthält 191 Westeuropäische Zeichen. **US-ASCII** (american standard code for information interchange) entspricht weitgehend dem Standard ISO-646 zur 7-bit Zeichenkodierung. Der Zeichensatz enthält 128 Zeichen und dient als Grundlage für auf mehr Bits basierende Kodierungen für Zeichensätze. **UTF** (*U*niversal Coded Character Set + *T*ransformation *F*ormat) hingegen ist eine Methode, Unicode-Zeichen auf Folgen von Bytes abzubilden. Der Unicode-Standard definiert UTF-32, UTF-16 und UTF-8 Kodierungen, welche alle Unicode-Zeichen kodieren können. **UTF-8** benützt 1 Byte um den ASCII Zeichensatz darzustellen und verwendet dieselbe Kodierung. Alle weiteren Zeichen des UTF werden mit UTF-8 nach Einsatzhäufigkeit mit variabler Länge kodiert. Dafür werden maximal 6 Bytes oder 48 Bits verwendet, davon sind maximal 32 Bit für Zeichen-Kodierungsinformationen und 16 Bit Metainformationen reserviert.

Bits of code point	First code point	Last code point	Bytes in sequence	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
7	U+0000	U+007F	1	0xxxxxxx					
11	U+0080	U+07FF	2	110xxxxx	10xxxxxx				
16	U+0800	U+FFFF	3	1110xxxx	10xxxxxx	10xxxxxx			
21	U+10000	U+1FFFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx		
26	U+200000	U+3FFFFFF	5	111110xx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	
31	U+4000000	U+7FFFFFFF	6	1111110x	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx

Examples [edit]

Consider the encoding of the Euro sign, €.

1. The Unicode code point for "€" is U+20AC.
2. According to the scheme table above, this will take three bytes to encode, since it is between U+0800 and U+FFFF.
3. Hexadecimal `20AC` is binary `0010 0000 1010 1100`. The two leading zeros are added because, as the scheme table shows, a three-byte encoding needs exactly sixteen bits from the code point.
4. Because the encoding will be three bytes long, its leading byte starts with three 1s, then a 0 (`1110...`)
5. The first 4 bits of the code point are stored in the remaining low order 4 bits of this byte (`1110 0010`), leaving 12 bits of the code point yet to be encoded (`...0000 1010 1100`).
6. All continuation bytes contain exactly 6 bits from the code point. So the next 6 bits of the code point are stored in the low order 6 bits of the next byte, and `10` is stored in the high order two bits to mark it as a continuation byte (so `1000 0010`).
7. Finally the last 6 bits of the code point are stored in the low order 6 bits of the final byte, and again `10` is stored in the high order two bits (`1010 1100`).

The three bytes `1110 0010` `1000 0010` `1010 1100` can be more concisely written in hexadecimal, as `E2 82 AC`.

Vor und Nachteile:

ASCII:

(+) Speicherplatzsparende 7 Bit Kodierung; optimiert für Englisch; dient als Basis für Latin-1 und UTF-8 Standard.

(-) Westeuropäische Zeichen fehlen; nicht häufig benutzt zur Kodierung von Websites (nur indirekt via UTF-8 und Latin-1); es werden 8 Bit benötigt für 7 Bit Kodierung.

Latin-1:

(+) Speicherplatzsparende 8 Bit Kodierung; optimiert für Mitteleuropäische Sprachen; Latin-1 ist mit rund 7% nach UTF-8 die zweithäufigste Kodierung von Websites; kommt man mit den 191 Zeichen zurecht, sollte man Latin 1 verwenden, da es aufgrund der festen Länge keine Probleme mit Anwendungen verursacht.

(-) Enthält nur 191 Zeichen; reicht nicht für alle europäischen Zeichen; gibt Probleme wenn eine Umgebung UTF-8 und Latin-1 gleichzeitig benutzt.

UTF-8:

(+) Mit Abstand häufigste Kodierung von Websites; kann alle möglichen 32 Bit UTF Zeichen kodieren und dekodieren; kodiert die Zeichen je nach Auftretenshäufigkeit mit variabler Länge und maximal 6 Bytes;
(-) Die maximal benötigten 6 Bytes enthalten 16 Bit Metainformationen dh. 2 Bytes werden maximal verschwendet; in einer seltener gebrauchten Sprache benötigen die Texte sehr viel Speicherplatz; obwohl mit UTF-8 jedes Zeichen kodiert werden kann, kann auch UTF-8 Nicht-UTF-8 kodierte Texte nicht oder nicht fehlerfrei encodieren, wenn diese ein anderes Bitmuster aufweisen. Einige Anwendungen wie beispielsweise mysql unterstützen nur maximal 3 Byte Kodierungen, somit gibt es bei seltenen Zeichen Probleme.

Reflexion/Feedback
a) Fasse deine Erkenntnisse und Lernfortschritte in zwei Sätzen zusammen.
Roland: Kennenlernen und anwenden der Regex Bibliothek re. Verstehen was der Unterschied ist zwischen dem UTF Zeichensatz, der UTF-8 Kodierung und der Latin-1 Kodierung.
b) Wie viel Zeit hast du in diese Übungen investiert?
Roland: Für die Aufgaben 1,2 und 3 etwa 12 Stunden. Allerdings war ich die meiste Zeit davon auf Wikipedia, Stackoverflow und der Python Dokumentation. Dann noch etwas Zeit, um die von Lennart gelöste Aufgabe 4 zu testen und nachzuvollziehen.