UNIVERSITÄT WIEN
CSLEARN - EDUCATIONAL TECHNOLOGIES

Natural Language Processing

Exercise Sheet 3

Processing Raw Text

Exercise 1

Rewrite the following loop as a list comprehension:

```
>>> sent = ['The', 'dog', 'gave', 'John', 'the', 'newspaper']
>>> result = []
>>> for word in sent:
...     word_len = (word, len(word))
...     result.append(word_len)
>>> result
[('The', 3), ('dog', 3), ('gave', 4), ('John', 4),
 ('the', 3), ('newspaper', 9)]
```

Exercise 2

Pig Latin is a simple transformation of English text. Each word of the text is converted as follows: move any consonant (or consonant cluster) that appears at the start of the word to the end, then append “ay”, e.g. “string” → “ingstray”. If a word starts with a vowel, just add “way” to the end, e.g. “idle” → “idleway”.

Write a function to convert a word to Pig Latin. Test it with the words “pig”, “cheers”, and “omelet”.

Exercise 3

Python’s `random` module includes a function `choice()` which randomly chooses an item from a sequence, e.g. `choice('aehh ')` will produce one of four possible characters, with the letter “h” being twice as frequent as the others. Write a generator expression that produces a sequence of 500 randomly chosen letters drawn from the string “aehh ”, and put this expression inside a call to the `''.join()` function, to concatenate them into one long string. You should get a result that looks like uncontrolled sneezing or maniacal laughter: “he haha ee heheeh eha”. Use `split()` and `join()` again to normalize the whitespace in this string.

Exercise 4

Readability measures are used to score the reading difficulty of a text, for the purposes of selecting texts of appropriate difficulty for language learners. Let us define μ_w to be the average number of letters per word, and μ_s to be the average number of words per sentence, in a given text. The Automated Readability Index (ARI) of the text is defined to be: $4.71\mu_w + 0.5\mu_s - 21.43$. Compute the ARI score for the “lore” and “learned” genre of the Brown Corpus. Make use of the fact that `nltk.corpus.brown.words()` produces a sequence of words, while `nltk.corpus.brown.sents()` produces a sequence of sentences.

Exercise 5

Define a variable `silly` to contain the string: 'newly formed bland ideas are inexpressible in an infuriating way'. Now write code to perform the following tasks:

a) Split `silly` into a list of strings, one per word, using Python's `split()` operation, and save this to a variable called `bland`.

b) Extract the second letter of each word in `silly` and join them into a string, to get 'eoldrnnnna'.

c) Combine the words in `bland` back into a single string, using `join()`. Make sure the words in the resulting string are separated with whitespace.

d) Print the words of `silly` in alphabetical order, one per line.

Exercise 6

Rewrite the following nested loop as a nested list comprehension:

```
>>> words = ['attribution', 'confabulation', 'tenacious', 'elocution',
...          'sequoia', 'tenacious', 'unidirectional']
>>> vsequences = set()
>>> for word in words:
...     vowels = []
...     for char in word:
...         if char in 'aeiou':
...             vowels.append(char)
...     vsequences.add(''.join(vowels))
>>> sorted(vsequences)
['aiuio', 'eaiou', 'eouio', 'euoia', 'oauaio', 'uiieioa']
```

Exercise 7

Write a program in SWI-Prolog that solves the task from the previous exercise. Some useful predicates are:

a) `atom_chars(Atom, CharList)`: transforms an atom into a list of characters and vice versa,

b) `memberchk(Elem, List)`: checks if `Elem` is an element of `List`,

c) `list_to_ord_set(List, OrdSet)`: transforms a list into an ordered set.