lexlib Documentation

Release 3.0.0

R. Steiner

Feb 08, 2019

Contents:

1 Input and output 1

2 Neighbors 3

3 Structure 5

4 Indices and tables 7

Python Module Index 9

CHAPTER 1

Input and output

Functions for reading and writing files.

lexlib.io.**get_words**(*file_path*, *column_name*, *delimiter=','*, ***fmtparams*)

Return a list containing only the items from the *column_name* column in the *delimiter*-separated file found at *file_path*. Also takes any of *csv.DictReader*'s *fmtparams*.

CHAPTER 2

Neighbors

Neighbor calculation functions for lexlib.

lexlib.neighbors.**check_neighbors**(*a*, *b*, *sep=None*)

Determine whether two words are neighbors. Returns *True* if they are neighbors and *False* if they are not.

sep – String used to separate phonemes (if the words are phonological forms). To separate into individual characters, set to *None* (default).

lexlib.neighbors.**get_neighbor_dict**(*words*, ***kwargs*)

Compare each word in a list of *words* to each word in a *corpus* word list (or in the same list if *corpus* is not given), and return a dict where each target word is a key, and its value is a list of its neighbors. (If you are looking for a function to get neighbor pairs, see *get_neighbor_pairs()*).

keyword arguments: *corpus* – List of all the words to get the neighbors from. If empty, defaults to *words*.

sep – String used to separate phonemes (if the words are phonological forms). To separate into individual characters, set to *None* (default).

debug – If *True*, it prints the current word and the words being compared to it to the console. Defaults to *False*.

lexlib.neighbors.**get_neighbor_pairs**(*words*, ***kwargs*)

Compare each word in a list of *words* to each word in a *corpus* word list (or in the same list if *corpus* is not given), and return a list of *(word, neighbor)* pairs. (If you are looking for a function to get lists of all the neighbors for specific words, see *get_neighbor_pairs()*).

keyword arguments: *corpus* – List of all the words to get the neighbors from. If omitted, defaults to *words*.

sep – String used to separate phonemes (if the words are phonological forms). To separate into individual characters, set to *None* (default).

debug – If True, it logs the current word and the words being compared to it to the console. Defaults to False.

lexlib.neighbors.**get_neighbor_positions**(*neighbor_pairs*, *sep=None*)

Given a list of *(word1, word2) neighbor_pairs*, return a list of *(word1, word2, position)* triples, where *position* is the position in the words where the neighbor relationship is formed. Note that this can only be calculated for pairs of substitution neighbors. If the words differ in length, *position* will be *-1*.

Example:

```
>>> neighbor_pairs = [("cat", "cap"), ("cat", "cut"), ("cat", "cast")]
>>> get_neighbor_positions(neighbor_pairs)
[("cat", "cap", 3), ("cat", "cut", 2), ("cat", "cast", -1)]
```

lexlib.neighbors.**get_neighbor_types**(*neighbor_dict*, *sep=None*)

Given a *neighbor_dict* (where a key is a "target" word and its value is a list of all of its neighbors), return a list of *(word1, word2, relationship)* triples, where *relationship* is one of "deletion," "addition," "substitution," or "unknown".

CHAPTER 3

Structure

Functions related to the structure of words.

lexlib.structure.**clusters**(*words*, *vowels*, *sep=None*, *unique=False*, *case_sensitive=True*)

Separates a list of *words* into clusters. Clusters are defined as sequences of characters that do not contain any of the characters in the list of *vowels*.

If *sep* is defined, it will be used as the delimiter string (for example, with *sep="."*, the word "a.bc.de" will be treated as the three-character sequence *["a", "bc", "de"]*).

If *unique* is *True*, returns each cluster only once. If *unique* is *False* (the default), returns each cluster as many times as it occurs.

If *case_sensitive* is *True* (the default), uppercase and lowercase characters will be treated as two different characters (e.g., "a" will be seen as different from "A"). If *case_sensitive* is *False*, uppercase and lowercase characters will be treated as the same character, and the output will be lowercase (e.g., "a" and "A" will both be treated as "a").

lexlib.structure.**clusters_word**(*word*, *vowels*, *sep=None*, *case_sensitive=True*)

Separates a *word* into clusters, defined as sequences of characters that do not contain any of the characters in the list of *vowels*.

If *sep* is defined, it will be used as the delimiter string (for example, with *sep="."*, the word "a.bc.de" will be treated as the three-character sequence *["a", "bc", "de"]*).

If *case_sensitive* is *True* (the default), uppercase and lowercase characters will be treated as two different characters (e.g., "a" will be seen as different from "A"). If *case_sensitive* is *False*, uppercase and lowercase characters will be treated as the same character, and the output will be lowercase (e.g., "a" and "A" will both be treated as "a").

lexlib.structure.**filter_by_nsyll**(*words*, *vowels*, *nsyll*, *sep=None*)

Given a list of *words*, return a list containing only the words with the desired number of syllables, determined by the number of characters from the *vowels* list found in that word.

The number of syllables, *nsyll* can be either an integer or a list of integers. If it is a list, the returned list will contain words of any syllable length included in *nsyll*.

If *sep* is defined, it will be used as the delimiter string (for example, with *sep="."*, the word "a.bc.de" will be treated as the three-character sequence *["a", "bc", "de"]*).

lexlib.structure.**get_cv**(*word*, *vowels*, *sep=None*)
Calculate the consonant ("C") and vowel ("V") structure of the given word. Returns a string of the characters "C" and "V" corresponding to the characters in the word.

vowels – A list of the characters representing vowels.

sep – String used to separate phonemes (if the words are phonological forms). To separate into individual characters, set to *None* (default).

lexlib.structure.**nsyll_list**(*words*, *vowels*, *sep=None*)
Count the number of syllables in each word in a *words* list, determined by the number of characters from the *vowels* list found in that word. Return a list of *(word, nsyll)* pairs.

If *sep* is defined, it will be used as the delimiter string (for example, with *sep="."*, the word "a.bc.de" will be treated as the three-character sequence *["a", "bc", "de"]*).

lexlib.structure.**nsyll_word**(*word*, *vowels*, *sep=None*)
Count the number of syllables in a *word*, determined by the number of characters from the *vowels* list found in that word.

If *sep* is defined, it will be used as the delimiter string (for example, with *sep="."*, the word "a.bc.de" will be treated as the three-character sequence *["a", "bc", "de"]*).

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

l

lexlib.io, 1
lexlib.neighbors, 3
lexlib.structure, 5

Index

C

check_neighbors() (*in module lexlib.neighbors*), 3
clusters() (*in module lexlib.structure*), 5
clusters_word() (*in module lexlib.structure*), 5

F

filter_by_nsyll() (*in module lexlib.structure*), 5

G

get_cv() (*in module lexlib.structure*), 6
get_neighbor_dict() (*in module lexlib.neighbors*), 3
get_neighbor_pairs() (*in module lexlib.neighbors*), 3
get_neighbor_positions() (*in module lexlib.neighbors*), 3
get_neighbor_types() (*in module lexlib.neighbors*), 4
get_words() (*in module lexlib.io*), 1

L

lexlib.io (*module*), 1
lexlib.neighbors (*module*), 3
lexlib.structure (*module*), 5

N

nsyll_list() (*in module lexlib.structure*), 6
nsyll_word() (*in module lexlib.structure*), 6