Package ‘keyToEnglish’

February 13, 2021

Type Package
Title Convert Data to Memorable Phrases
Version 0.2.1
Author Max Candocia
Maintainer Max Candocia <maxcandocia@gmail.com>
URL https://github.com/mcandocia/keyToEnglish
Description Convert keys and other values to memorable phrases.
   Includes some methods to build lists of words.
License GPL (>= 2)
Encoding UTF-8
LazyData true
Imports openssl, stringr, jsonlite
Depends R (>= 3.4.0)
RoxygenNote 7.1.1
Suggests testthat
NeedsCompilation no
Repository CRAN
Date/Publication 2021-02-13 17:50:02 UTC

R topics documented:
corpora_to_word_list ............................................... 2
GCD ................................................................. 3
generate_random_sentences ........................................ 4
hash_to_sentence .................................................. 4
keyToEnglish ....................................................... 5
LCM ................................................................. 6
reconcile_misspellings ............................................. 7
uniqueness_max_size ............................................... 7
uniqueness_probability ............................................ 8
**corpora_to_word_list**

Wiki clean ................................................. 8
wl_adjectives_nonorigin ................................. 9
wl_adjectives_visual .................................... 9
wl_animal .................................................. 10
wl_common .................................................. 10
wl_freq5663 ............................................. 11
wl_literature ............................................. 11
wl_nouns_concrete ....................................... 12
wl_nouns_concrete_plural .............................. 12
wl_science ............................................... 13
wl_verbs_transitive gerund ............................ 13
wl_verbs_transitive_infinitive ....................... 14
wl_verbs_transitive_present ........................... 14
wml_animals .............................................. 14
wml_cutephysics ......................................... 15
wml_long_sentence ....................................... 15

**Index**

16

---

**corpora_to_word_list  Corpore to Word List**

**Description**

Converts a collection of documents to a word list

**Usage**

```r
corpora_to_word_list(
  paths,
  ascii_only = TRUE,
  custom_regex = NA,
  max_word_length = 20,
  stopword_fn = DEFAULT_STOPWORDS,
  min_word_count = 5,
  max_size = 16^3,
  min_word_length = 3,
  output_file = NA,
  json_path = NA
)
```

**Arguments**

- **paths**
  - Paths of plaintext documents
- **ascii_only**
  - Will omit non-ascii characters if TRUE
- **custom_regex**
  - If not NA, will override ascii_only and this will determine what a valid word consists of
max_word_length  Maximum length of extracted words
stopword_fn  Filename containing stopwords to use or a list of stopwords (if length > 1)
min_word_count  Minimum number of occurrences for a word to be added to word list
max_size  Maximum size of list
min_word_length  Minimum length of words
output_file  File to write list to
json_path  If input text is JSON, then it will be parsed as such if this is a character of JSON keys to follow

Value

A ‘character’ vector of words

GCD  *Greater Common Denominator*

Description

Calculates greatest common denominator of a list of numbers

Usage

GCD(...)  

Arguments

...  Any number of ‘numeric’ vectors or nested ‘list’s containing such

Value

A ‘numeric’ that is the greatest common denominator of the input values
generate_random_sentences

*Generate Random Sentences*

**Description**
Randomly generate sentences with a specific structure

**Usage**
```
generate_random_sentences(n, punctuate = TRUE, fast = FALSE)
```

**Arguments**
- `n`: 'numeric' number of sentences to generate
- `punctuate`: 'logical' value of whether to add spaces, capitalize first letter, and append period
- `fast`: 'logical'

**Value**
'character' vector of randomly generated sentences

---

hash_to_sentence

*Hash to Sentence*

**Description**
Hashes data to a sentence that contains 54 bits of entropy

**Usage**
```
hash_to_sentence(x, ...)
```

**Arguments**
- `x`: Input data, which will be converted to 'character' if not already 'character'
- `...`: Other parameters to pass to `keyToEnglish()`, besides 'word_list', 'hash_subsection_size', and 'hash_function'

**Value**
'character' vector of hashed field resembling phrases
keyToEnglish

Key to English

Description

Hashes field to sequence of words from a list.

Usage

```r
keyToEnglish(
  x,
  hash_function = "md5",
  phrase_length = 5,
  corpus_path = NA,
  word_list = wl_common,
  hash_subsection_size = 3,
  sep = "",
  word_trans = "camel",
  suppress_warnings = FALSE,
  hash_output_length = NA,
  forced_limit = NA,
  numeric_append_range = NA
)
```

Arguments

- **x** - field to hash
- **hash_function** - 'character' name of hash function or 'function' itself, returning a hexadecimal character
- **phrase_length** - 'numeric' of words to use in each hashed key
- **corpus_path** - 'character' path to word list, as a single-column text file with one word per row
- **word_list** - 'character' list of words to use in phrases
- **hash_subsection_size** - 'numeric' length of each subsection of hash to use for word index. 16^N unique words can be used for a size of N. This value times phrase_length must be less than or equal to the length of the hash output. Must be less than 14.
- **sep** - 'character' separator to use between each word.
- **word_trans** - A 'function', 'list' of functions, or 'camel' (for CamelCase). If a list is used, then the index of the word of each phrase is mapped to the corresponding function with that index, recycling as necessary
- **suppress_warnings** - 'logical' value indicating if warning of non-character input should be suppressed
- **hash_output_length** - optional 'numeric' if the provided hash function is not a 'character'. This is used to send warnings if the hash output is too small to provide full range of all possible combinations of outputs.
for multiple word lists, this is the maximum number of values used for calculating the index (prior to taking the modulus) for each word in a phrase. Using this may speed up processing longer word lists with a large least-common-multiple among individual word list lengths. This will introduce a small amount of bias into the randomness. This value should be much larger than any individual word list whose length is not a factor of this value.

numeric_append_range

optional ‘numeric’ value of two integers indicating range of integers to append onto data

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘character’ vector of hashed field resembling phrases</td>
</tr>
</tbody>
</table>

**Examples**

```r
# hash the numbers 1 through 5
df$test <- keyToEnglish(1:5)

# alternate upper and lowercase, 3 words only
df$test <- keyToEnglish(1:5, word_trans=list(tolower, toupper), phrase_length=3)
```

---

### LCM

**Least Common Multiple**

**Description**

Calculates least common multiple of a list of numbers

**Usage**

```r
LCM(...) 
```

**Arguments**

```r
... Any number of ‘numeric’ vectors or nested ‘list’s containing such
```

**Value**

A ‘numeric’ that is the least common multiple of the input values
reconcile_misspellings

Reconcile Misspellings

Description

Fixes incorrect spellings from previous versions of this package

Usage

reconcile_misspellings(x, from = "0.2.0", to = "0.2.1")

Arguments

x 'character' vector of phrases from 'keyToEnglish()' output or related function
from 'character' version number of how the hash was originally generated
to 'character' version that hash should be converted to

Value

'character' vector of fixed phrase hash compatible with output for version 'to'

uniqueness_max_size

Uniqueness Max Size

Description

Returns approximate number of elements that you can select out of a set of size 'N' if the probability of there being any duplicates is less than or equal to 'p'

Usage

uniqueness_max_size(N, p)

Arguments

N 'numeric' size of set elements are selected from, or a 'list' of 'list's of 'character' vectors (e.g., 'wml_animals')
p 'numeric' probability that there are any duplicate elements

Value

'numeric' value indicating size. Value will most likely be non-integer
Examples
# how many values from 1-1,000 can I randomly select before
# I have a 10% chance of having at least one duplicate?

uniqueness_max_size(1000, 0.1)
# 14.51

uniqueness_probability

Uniqueness Probability

Description
Calculates probability that all ‘r’ elements of a set of size ‘N’ are unique

Usage
uniqueness_probability(N, r)

Arguments
N       'numeric' size of set. Becomes unstable for values greater than 10^16.
r       'numeric' number of elements selected with replacement

Value
'numeric' probability that all ‘r’ elements are unique

wiki_clean

Clean JSON text from Wikipedia

Description
Clean JSON text from Wikipedia

Usage
wiki_clean(x)

Arguments
x       'character' JSON text

Value
'character' JSON text
**wl_adjectives_nonorigin**

*Non-Origin Adjectives Wordlist*

**Description**

Word list of 256 adjectives that do not describe origin, so they can usually be used prior to visual/origin adjectives without breaking any grammar rules.

**Usage**

```r
data(wl_adjectives_nonorigin)
```

**Format**

A `character` vector

---

**wl_adjectives_visual**  
*Visual Adjectives Wordlist*

**Description**

Word list of 256 adjectives that visually describe an object.

**Usage**

```r
data(wl_adjectives_visual)
```

**Format**

A `character` vector
### wl_animal

**Animal word list**

**Description**
Word list generated by processing several animal-related pages on Wikipedia

**Usage**
data(wl_animal)

**Format**
An object of class 'character'

**Examples**
data(wl_animal)
keyToEnglish(1:5, word_list=wl_animal)

### wl_common

**Common word list**

**Description**
Public domain word list of common words

**Usage**
data(wl_common)

**Format**
An object of class 'character'

**References**

**Examples**
data(wl_common)
keyToEnglish(1:5, word_list=wl_common)
**Freq 5663 word list**

**Description**
Public domain word list of common words, slightly truncated from original version

**Usage**
```r
data(wl_freq5663)
```

**Format**
An object of class `character`

**References**

**Examples**
```r
data(wl_common)
keyToEnglish(1:5, word_list=wl_freq5663)
```

**Literature word list**

**Description**
Word list generated by processing several works of literature on Project Gutenberg

**Usage**
```r
data(wl_literature)
```

**Format**
An object of class `character`

**References**
Project Gutenberg. [Project Gutenberg](https://project-gutenberg.org)

**Examples**
```r
data(wl_literature)
keyToEnglish(1:5, word_list=wl_literature)
```
Concrete Nouns Wordlist

Description

Word list of 2048 singular, concrete nouns, largely excluding materials and liquids that cannot be referred to in the singular form.

Usage

data(wl_nouns_concrete)

Format

A ‘character’ vector

Plural Concrete Nouns Wordlist

Description

Word list of 2048 concrete nouns in plural form, largely excluding materials and liquids that cannot be referred to in the singular form.

Usage

data(wl_nouns_concrete_plural)

Format

A ‘character’ vector
**Description**

Word list generated by processing several science-related pages on Wikipedia

**Usage**

data(wl_science)

**Format**

An object of class 'character'

**Examples**

data(wl_science)
keyToEnglish(1:5, word_list=wl_science)

---

**wl_verbs_transitive_gerund**  
*Transitive Verbs in Gerund Form*

**Description**

Word list of 256 transitive verbs in gerund form (i.e., "ing" at end)

**Usage**

data(wl_verbs_transitive_gerund)

**Format**

A 'character' vector
**wml_animals**

**Transitive Verbs in Infinitive Form**

**Description**

Word list of 256 transitive verbs in infinitive form (minus the "to")

**Usage**

`data(wl_verbs_transitive_infinitive)`

**Format**

A 'character' vector

**Transitive Verbs in Present Form**

**Description**

Word list of 256 transitive verbs in present tense

**Usage**

`data(wl_verbs_transitive_present)`

**Format**

A 'character' vector

**Animal Phrase Structure Word Multilist**

**Description**

Word lists of sizes, colors, animals, and attributes to construct memorable phrases

List of word lists that combine cute words with physics-related words

**Usage**

`data(wml_animals)`

`data(wml_animals)`
**wml_cutephysics**

**Format**
- A ‘list’ of ‘character’ vectors
- A ‘list’ of ‘character’ vectors

**Examples**

```
keyToEnglish(1:5, word_list=wml_animals)
```

---

**Description**

List of word lists that combine cute words with physics-related words.

**Usage**

```
data(wml_cutephysics)
```

**Format**

A ‘list’ of ‘character’ vectors

---

**wml_long_sentence**

**Long Sentence Multilist**

**Description**

List of word lists that can be used to make a 54-byte, often humorous, sentence.

**Usage**

```
data(wml_long_sentence)
```

**Format**

A ‘list’ of ‘character’ vectors
Index

* datasets
  wl_adjectives_nonorigin, 9
  wl_adjectives_visual, 9
  wl_animal, 10
  wl_common, 10
  wl_freq5663, 11
  wl_literature, 11
  wl_nouns_concrete, 12
  wl_nouns_concrete_plural, 12
  wl_science, 13
  wl_verbs_transitive_gerund, 13
  wl_verbs_transitive_infinitive, 14
  wl_verbs_transitive_present, 14
  wml_animals, 14
  wml_cutephysics, 15
  wml_long_sentence, 15

  corpora_to_word_list, 2
  GCD, 3
  generate_random_sentences, 4
  hash_to_sentence, 4
  keyToEnglish, 5
  LCM, 6
  reconcile_misspellings, 7
  uniqueness_max_size, 7
  uniqueness_probability, 8
  wiki_clean, 8
  wl_adjectives_nonorigin, 9
  wl_adjectives_visual, 9
  wl_animal, 10
  wl_common, 10
  wl_freq5663, 11
  wl_literature, 11
  wl_nouns_concrete, 12