Package ‘ngram’

February 20, 2015

Type Package
Title An n-gram Babbler
Version 1.1
Description This package offers utilities for creating, displaying, and "babbling" n-grams. The babbler is a simple Markov process.
License BSD 2-clause License + file LICENSE
Depends R (>= 2.14.0), methods
LazyData yes
LazyLoad yes
NeedsCompilation yes
ByteCompile yes
Author Drew Schmidt [aut, cre], Christian Heckendorf [aut]
Maintainer Drew Schmidt <wrathematics@gmail.com>
Repository CRAN
Date/Publication 2014-06-25 07:02:26

R topics documented:

  ngram-package .................................................. 2
  Babble .......................................................... 2
  ngram Getters ................................................... 3
  ngram print ...................................................... 4
  ngram-class ...................................................... 5
  Preprocess ..................................................... 6
  Process ......................................................... 7
  Utilities ......................................................... 8

Index 10
ngram-package

ngram: An n-gram Babbler

Description

This package offers utilities for creating, displaying, and "babbling" n-grams. The babbler is a simple Markov process.

Details

The ngram package is distributed under the permissive 2-clause BSD license. If you find the code here useful, please let us know and/or cite the package, whatever is appropriate.

The package has its own PRNG; we use an implementation of MT1997 for all non-deterministic choices.

Author(s)

Drew Schmidt <wramatics AT gmail.com>, Christian Heckendorf.

Babble

ngram Getters

Description

A markov chain babbler.

Usage

```r
## S4 method for signature 'ngram'
babble(ng, genlen=150, seed=getseed())
```

Arguments

- `ng`: An ngram object.
- `genlen`: Generated length, i.e., the number of words to babble.
- `seed`: Seed for the random number generator.

Details

The babbler uses its own internal PRNG (i.e., not R’s), so seeds cannot be managed as with R’s seeds. The generator is an implementation of MT19937.

Methods

```r
signature(ng = "ngram")
```


### ngram Getters

**See Also**

ngram

**Examples**

```r
# Not run:
library(ngram)

str <- "A B A C A B B"
ng <- ngram(str)
babble(ng, genlen=5, seed=1234)

# End(Not run)
```

### Description

Some simple "getters" for ngram objects. Necessary since the internal representation is not a native R object.

### Usage

```r
# S4 method for signature 'ngram'
get.ngrams(ng)

# S4 method for signature 'ngram'
get.string(ng)

# S4 method for signature 'ngram'
get.nextwords(ng)
```

### Arguments

- **ng**: An ngram object.

### Details

- `get.ngrams()` returns an R vector of all n-grams.
- `get.nextwords()` does nothing at the moment; it will be implemented in future releases.
- `getnstring()` recovers the input string as an R string.

### Methods

- `signature(ng = "ngram")`
See Also

ngram

Examples

## Not run:
library(ngram)

str <- "A B A C A B B"
ng <- ngram(str)
get.ngrams(ng)

## End(Not run)
Description

An n-gram is an ordered sequence of n "words" taken from a body of "text". The terms "words" and "text" can easily be interpreted literally, or with a more loose interpretation.

For example, consider the sequence "A B A C A B B". If we examine the 2-grams (or bigrams) of this sequence, they are


or without repetition:

A B, B A, A C, C A, B B

That is, we take the input string and group the "words" 2 at a time (because \( n = 2 \)). Notice that the number of n-grams and the number of words are not obviously related; counting repetition, the number of n-grams is equal to

\[ n_{\text{words}} - n + 1 \]

Bounds ignoring repetition are highly dependent on the input. A correct but useless bound is

\[ \# \text{ngrams} = \# \text{nwords} - (\# \text{repeats} - 1) - (n - 1) \]

An ngram object is an S4 class container that stores some basic summary information (e.g., n), and several external pointers. For information on how to construct an ngram object, see ngram.

Creating Objects

\[
\text{new}(\text{'ngram'}, \text{str_ptr} = \ldots, \text{strlen} = \ldots, n = \ldots, \text{ng_ptr} = \ldots, \text{ngsize} = \ldots, \text{wl_ptr} = \ldots)
\]

Slots

- str_ptr: An external pointer to a copy of the input string.
- strlen: The number of characters of the input string.
- n: The canonical 'n', as in 'n-gram'.
- ng_ptr: External pointer to the list of n-grams.
- ngsize: The total number of n-grams
- wl_ptr: External pointer to the list of words.

Details

The ngram class is a container for the output of the processing routine ngram(), most of which are external pointers. As such, does not store much data (a few KiB), regardless of the input data size. Additionally, this makes saving such objects via save() and then loading them later with load() useless at best, and dangerous at worst.

See Also

Process
Preprocess

Preprocessing

Description

A simple text preprocessor for use with the \texttt{ngram()} function.

Usage

\begin{verbatim}
preprocess(x, case=NULL, split.at.punct=FALSE)
\end{verbatim}

Arguments

- \texttt{x}: Input text.
- \texttt{case}: Option to change the case of the text. See Details section for appropriate values.
- \texttt{split.at.punct}: logical; determines if spaces should be inserted before and after punctuation (making them individual characters for an n-gram model).

Details

The input text \texttt{x} must already be in the correct form for \texttt{ngram()}, i.e., a single string (character vector of length 1).

The \texttt{case} argument can take 3 possible values: \texttt{NULL}, in which case nothing is done, or \texttt{lower} or \texttt{upper}, wherein the case of the input text will be made lower/upper case, respectively.

Value

\texttt{concat()} returns

See Also

\texttt{Process}, \texttt{Utilities}

Examples

\begin{verbatim}
## Not run:
library(ngram)

x <- "Watch out for snakes!"
p = preprocess(x)
p = preprocess(x, case="upper", split.at.punct=TRUE)

## End(Not run)
\end{verbatim}
Description

This routine processes a string into an ngram class object.

Usage

```r
## S4 method for signature 'character'
ng(x, n=2)
```

Arguments

- `x` The input text.
- `n` The 'n' as in 'n-gram'.

Details

The `ngram()` function is the main workhorse of this package. It takes an input string and converts it into the internal n-gram representation.

On evaluation, a copy of the input string is produced and stored as an external pointer. This is necessary because the internal list representation just points to the first char of each word in the input string. So if you (or R's gc) deletes the input string, basically all hell breaks loose.

Methods

```r
signature(x = "character")
```

See Also

`ngram-class`

Examples

```r
## Not run:
library(ngram)

str <- "A B A C A B B"
ng <- ngram(str, n=2)
ng

## End(Not run)
```
**Utilities**

- **n-gram Utilities**

**Description**

Some utility methods.

**Usage**

```r
concat(..., collapse="", rm.space=FALSE)
```

```r
## S4 method for signature 'character'
wordcount(x)
```

```r
## S4 method for signature 'ngram'
wordcount(x)
```

**Arguments**

- `...` Input text(s).
- `x` A string or vector of strings.
- `collapse` A character to separate the input strings if a vector of strings is supplied; otherwise this does nothing.
- `rm.space` logical; determines if spaces should be removed from the final string.

**Details**

`concat()` is a utility for concatenating strings together. This is handy because if you want to generate the n-grams for several different texts, you must first put them into a single string.

`wordcount()` counts words. Currently a "word" is a clustering of characters separated from another clustering of characters by at least 1 space. That is the law.

**Value**

`concat()` returns

**Methods**

```r
signature(x = "character")
```

```r
signature(x = "ngram")
```

**See Also**

`Preprocess`
Examples

```r
## Not run:
library(ngram)

words <- c("a", "b", "c")
wordcount(words)
str <- concat(words)
wordcount(str)

## End(Not run)
```
# Index

<table>
<thead>
<tr>
<th>Topic</th>
<th>Class</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Generation</td>
<td>Process</td>
<td>7</td>
</tr>
<tr>
<td>Package</td>
<td>ngram-package</td>
<td>2</td>
</tr>
<tr>
<td>Babble</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>babble</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>babble(Babble)</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>babble, ngram-method(Babble)</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>concat</td>
<td>(Utilities)</td>
<td>8</td>
</tr>
<tr>
<td>get.nextwords</td>
<td>(ngram Getters)</td>
<td>3</td>
</tr>
<tr>
<td>get.nextwords-method(ngram Getters)</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>get.ngrams</td>
<td>(ngram Getters)</td>
<td>3</td>
</tr>
<tr>
<td>get.ngrams-method(ngram Getters)</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>get.ngrams-method(ngram Getters)</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>get.string</td>
<td>(ngram Getters)</td>
<td>3</td>
</tr>
<tr>
<td>get.string-method(ngram Getters)</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>ngram</td>
<td></td>
<td>3–5</td>
</tr>
<tr>
<td>ngram(Process)</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>ngram Getters</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>ngram print</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>ngram.character-method(Process)</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>ngram-class</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>ngram-method</td>
<td>(Process)</td>
<td>7</td>
</tr>
<tr>
<td>ngram-package</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Preprocess</td>
<td></td>
<td>6, 8</td>
</tr>
<tr>
<td>preprocess</td>
<td>(Preprocess)</td>
<td>6</td>
</tr>
<tr>
<td>print</td>
<td>(ngram print)</td>
<td>4</td>
</tr>
<tr>
<td>print,ngram-method(ngram print)</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>show</td>
<td>(ngram print)</td>
<td>4</td>
</tr>
<tr>
<td>show,ngram-method(ngram print)</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>show-method</td>
<td>(ngram print)</td>
<td>4</td>
</tr>
<tr>
<td>Utilities</td>
<td></td>
<td>6, 8</td>
</tr>
<tr>
<td>wordcount</td>
<td>(Utilities)</td>
<td>8</td>
</tr>
<tr>
<td>wordcount,character-method(Uutilities)</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>wordcount,ngram-method(Uutilities)</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>wordcount-method(Uutilities)</td>
<td></td>
<td>8</td>
</tr>
</tbody>
</table>