Markovman

Generated by Doxygen 1.8.13

Contents

Chapter 1

Main Page

Implementation of markov chains for random text generation.

1.1 Description

Markovman is a program for random text generation based on markov chains. The generator is trained from a corpus. The only supported format for the corpus is as a text file, with dots '.' separating sentences.

1.2 Usage

The following is the interface as I plan to implement it, although it hasn't been written yet. The easiest way to use Markovman is to call it together with a corpus-file.

```
markovman path/to/corpus.txt
```

That will put the program in a loop, reading from stdin. You can pass the following commands:

```
gen N
```

will generate N sentences one after the other based on the corpus.

```
kill X
```

will make the word X disappear from the corpus.

```
exit
```

will exit the program

Another possibility is running the program like the following, which will generate N sentences and close immediately.

```
markovman path/to/corpus.txt -n N
```

See also

https://github.com/IanTayler/markovman.git

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 ThisWord Struct Reference

Data Fields

- struct ThisWord ** **wordlist**
- int * **freqlist**
- int **length**

The documentation for this struct was generated from the following file:

- src/lib/statemach.c

4.2 Word Struct Reference

Struct for representing states in a first order Markov chain.

4.2.1 Detailed Description

Struct for representing states in a first order Markov chain.

The struct consists of:

- wordlist: a pointer to an array of pointers to other Words (the possible follow-ups)
- freqlist: a pointer to an array of integers. Marks the frequency of each item in wordlist.
- length: the length of both the above arrays.

The documentation for this struct was generated from the following file:

- src/lib/statemach.c

Chapter 5

File Documentation

5.1 src/include/minunit.h File Reference

A very minimal unit test library.

Macros

- #define mu_assert(message, test) do { if (!(test)) return message; } while (0)

    Macro to assert equality in a unit test.
- #define mu_run_test(test)

    Macro to run a test.

Variables

- int tests_run

    Global set to the amount of tests that ran.

5.1.1 Detailed Description

A very minimal unit test library.

Author

Jera Design

Date

Unknown

See also

`http://www.jera.com/techinfo/jtns/jtn002.html`

5.1.2 Macro Definition Documentation

5.1.2.1 mu_assert

```
#define mu_assert(
              message,
              test ) do { if (!(test)) return message; } while (0)
```

Macro to assert equality in a unit test.

This macro checks whether 'test' is a true value. If it is, then the macro does nothing. Otherwise, it will pass a message as the return value of the function in which the macro will be expanded.

Parameters

message	This message will be the return value of whichever function implements mu_assert. It should be a message to be sent if the assertion fails.
test	This is the value being asserted. It should evaluate to a true value in successful tests.

5.1.2.2 mu_run_test

```
#define mu_run_test(
              test )
```

Value:

```
do { char *message = test(); tests_run++; \
                                if (message) return message; } while (0)
```

Macro to run a test.

This macro is used to run a 'test' function, which should return 0 if everything is alright. This macro should be included in functions with a ∗char return type.

Parameters

test	A pointer to a function that resturns 0 if everything is alright and a message (∗char) if there's an error.

5.1.3 Variable Documentation

5.1.3.1 tests_run

```
int tests_run
```

Global set to the amount of tests that ran.

This variable gets increased when mu_run_test runs, and it should hold the amount of tests ran at the end of the test program.

See also

mu_run_test

5.2 src/include/statemach.h File Reference

Header file for state machines.

5.2.1 Detailed Description

Header file for state machines.

Author

Ian G. Tayler

Date

5 May 2017 (creation)

This exports the names from lib/statemach.c that we will need in src/main.c.

See also

`https://github.com/IanTayler/markovman.git`

5.3 src/lib/statemach.c File Reference

File implementing state machines.

Data Structures

- struct ThisWord

Typedefs

- typedef struct ThisWord **Word**

5.3.1 Detailed Description

File implementing state machines.

Author

Ian G. Tayler

Date

5 May 2017 (creation)

This is the file where all the action happens. We define the struct 'Word' and a few functions for handling it. That covers most of the program's logic.

See also

`https://github.com/IanTayler/markovman.git`

5.4 src/markovman.c File Reference

The main file, where the interface is implemented.

```
#include <stdio.h>
#include "statemach.h"
```

Macros

- #define VERSION "0.0.2"

  String constant holding the current version of Markovman.

Functions

- int **main** (void)

5.4.1 Detailed Description

The main file, where the interface is implemented.

Author

Ian G. Tayler

Date

5 May 2017 (creation)

See also

`https://github.com/IanTayler/markovman.git`

Index