

API Documentation

API Documentation

November 29, 2010

Contents

Contents	1
1 Package bard	2
1.1 Modules	2
1.2 Functions	2
1.3 Variables	2
2 Package bard.detokenizers	3
2.1 Modules	3
2.2 Functions	3
3 Module bard.detokenizers.regex	4
3.1 Variables	4
3.2 Class RegexDetokenizer	4
3.2.1 Methods	4
4 Module bard.detokenizers.tests	5
4.1 Class TestDetokenizer	5
4.1.1 Methods	5
4.1.2 Properties	5
5 Package bard.generators	7
5.1 Modules	7
5.2 Functions	7
6 Module bard.generators.markov	8
6.1 Variables	8
6.2 Class MarkovGenerator	8
6.2.1 Methods	8
6.3 Class IntelligentMarkovGenerator	9
6.3.1 Methods	10
7 Module bard.generators.sentence	11
7.1 Variables	11
7.2 Class SentenceBasedGenerator	11
7.2.1 Methods	11
8 Module bard.generators.tests	12

8.1	Variables	12
8.2	Class TestMarkov	12
8.2.1	Methods	12
8.2.2	Properties	13
8.3	Class TestIntelligentMarkov	13
8.3.1	Methods	13
8.3.2	Properties	13
8.4	Class TestSentenceBased	14
8.4.1	Methods	14
8.4.2	Properties	14
9	Module bard.test	16

1 Package bard

A module which generates pseudorandom text.

It utilizes Markov chains to produce new text based on some input text. You could use it, for example, to write a ten million word science fiction epic using the science fiction category of the Brown corpus (yes, I have done it).

This module requires NLTK.

Version: 0.2

Author: Zach Denton

1.1 Modules

- **detokenizers** (*Section 2, p. 3*)
 - **regex** (*Section 3, p. 4*)
 - **tests** (*Section 4, p. 5*)
- **generators** (*Section 5, p. 7*)
 - **markov** (*Section 6, p. 8*)
 - **sentence** (*Section 7, p. 11*)
 - **tests** (*Section 8, p. 12*)
- **test** (*Section 9, p. 16*)

1.2 Functions

<code>generate_text(length=100)</code>
--

1.3 Variables

Name	Description
<code>--package--</code>	Value: 'bard'

2 Package *bard.detokenizers*

2.1 Modules

- `regex` (Section 3, p. 4)
- `tests` (Section 4, p. 5)

2.2 Functions

`detokenize(tokens)`

detokenize tokens using the currently-recommended detokenizer

```
>>> tokens = ['', 'What', '...', 'is', 'the', 'airspeed', 'velocity', 'of', 'an', 'unladen', 'swallow']
>>> sentence = detokenize(tokens)
>>> print sentence
'What... is the airspeed velocity of an unladen swallow?'
<BLANKLINE>
'African or European?'
<BLANKLINE>
'I don't know that!'
```

3 Module `bard.detokenizers.regex`

3.1 Variables

Name	Description
<code>--package--</code>	Value: <code>'bard.detokenizers'</code>

3.2 Class `RegexDetokenizer`

3.2.1 Methods

<code>detokenize(self, tokens)</code>
A regex-based detokenizer.
Pass a list of tokens, and you will receive a string formatted like a novel. Dialogue is placed on its own line.

4 Module *bard.detokenizers.tests*

4.1 Class *TestDetokenizer*



4.1.1 Methods

setUp(*self*)

Hook method for setting up the test fixture before exercising it.

Overrides: *unittest.TestCase.setUp* extit(inherited documentation)

test_regex(*self*)

test_regex_tagged(*self*)

test_default(*self*)

test_default_tagged(*self*)

Inherited from unittest.TestCase

__call__(), *__eq__()*, *__hash__()*, *__init__()*, *__ne__()*, *__repr__()*, *__str__()*, *assertAlmostEqual()*, *assertAlmostEquals()*, *assertEqual()*, *assertEquals()*, *assertFalse()*, *assertNotAlmostEqual()*, *assertNotAlmostEquals()*, *assertNotEqual()*, *assertNotEquals()*, *assertRaises()*, *assertTrue()*, *assert_()*, *countTestCases()*, *debug()*, *defaultTestResult()*, *fail()*, *failIf()*, *failIfAlmostEqual()*, *failIfEqual()*, *failUnless()*, *failUnlessAlmostEqual()*, *failUnlessEqual()*, *failUnlessRaises()*, *id()*, *run()*, *shortDescription()*, *tearDown()*

Inherited from object

__delattr__(), *__format__()*, *__getattr__()*, *__new__()*, *__reduce__()*, *__reduce_ex__()*, *__setattr__()*, *__sizeof__()*, *__subclasshook__()*

4.1.2 Properties

Name	Description
<i>Inherited from object</i>	
<i>__class__</i>	

continued on next page

Name	Description
------	-------------

5 Package `bard.generators`

5.1 Modules

- `markov` (*Section 6, p. 8*)
- `sentence` (*Section 7, p. 11*)
- `tests` (*Section 8, p. 12*)

5.2 Functions

<code>generate(<i>corpus</i>=None, <i>length</i>=100)</code>
use the best text generator to generate some pseudorandom text

6 Module `bard.generators.markov`

6.1 Variables

Name	Description
<code>__package__</code>	Value: <code>'bard.generators'</code>

6.2 Class `MarkovGenerator`

Known Subclasses: `bard.generators.markov.IntelligentMarkovGenerator`

Uses a Markov chain to generate random text from a list of tokens.

The tokens can be POS-tagged (a list of tuples) or not (a list of strings).

6.2.1 Methods

`__init__(self, tokens, use_cache=False)`

Initializes the `MarkovGenerator`.

If `use_cache` is `True`, the `MarkovGenerator` will attempt to use a pickled version of the trigram index. This provides a performance benefit on large corpora (such as the entire Brown corpus) but is slightly slower with smaller corpora (such as the science fiction category of the Brown corpus).

`generate(self, w1=None, w2=None, length=100)`

A pure version of the pseudorandom Markov chain text generator

Keyword arguments: `w1` – starting word `w2` – second word `length` – number of tokens to produce

This version does not have any additional intelligence, so it will produce illogical sentences. However, it will always produce the correct length.

```
>>> tokens = nltk.corpus.brown.tagged_words(categories="fiction")
>>> m = MarkovGenerator(tokens)
>>> text = m.generate(length=100)
>>> isinstance(text, str)
True
```

get_next (<i>self</i> , <i>w1</i> , <i>w2</i> , <i>search_for</i> , <i>exclude</i> =[])

find a trigram of the form (w1, w2, search_for)

get_largest (<i>self</i>)

return the key of the item in the cache with the most possibilities

get_starter (<i>self</i>)

return the key of the item in the cache which is best suited for starting the text.

get_random (<i>self</i>)

return a random item.

istagged (<i>self</i>)

determine whether our tokens are part-of-speech tagged or not

get_tags (<i>self</i>)

return the different part-of-speech tags in the cache

6.3 Class *IntelligentMarkovGenerator*

bard.generators.markov.MarkovGenerator

└─ bard.generators.markov.IntelligentMarkovGenerator

6.3.1 Methods

generate(*self*, *w1=None*, *w2=None*, *length=100*)

An enhanced version of the Markov chain text generator

Keyword arguments: *w1* – starting word *w2* – second word *length* – try to produce this many tokens

Contains some rules to ensure that the resultant text is logical, such as trying to close quotations and parentheses and not inserting quotations where they don't make sense. However, this is not always possible and thus there will still be some misplaced quotation marks and parentheses. Furthermore, this function will not stop producing text until it is satisfied that parentheses and quotations have been closed and the last character marks the end of a sentence.

```
>>> tokens = nltk.corpus.brown.tagged_words(categories='fiction')
>>> m = IntelligentMarkovGenerator(tokens)
>>> text = m.generate(length=100)
>>> isinstance(text, str)
True
```

Overrides: *bard.generators.markov.MarkovGenerator.generate*

*Inherited from **bard.generators.markov.MarkovGenerator**(Section 6.2)*

`__init__()`, `get_largest()`, `get_next()`, `get_random()`, `get_starter()`, `get_tags()`, `istagged()`

7 Module `bard.generators.sentence`

7.1 Variables

Name	Description
<code>__package__</code>	Value: <code>'bard.generators'</code>

7.2 Class `SentenceBasedGenerator`

This generator generates text using a model of the input tokens' sentence structure.

Basically it creates an index of the sentence structures in the tokens using their part-of-speech tags. It then maintains a separate index of all the part-of-speech tags in the text and the words that have those tags. Finally it picks a sentence from the sentence index and replaces the part-of-speech tags in the sentence with a random word which has that tag.

7.2.1 Methods

<code>__init__(self, tokens)</code>
--

Initialize the generator. tokens must be a list of pos-tagged sentences.
--

<code>generate(self, length=100)</code>
--

<code>get_word(self, tag)</code>

<code>get_sentence(self)</code>
--

<code>istagged(self)</code>

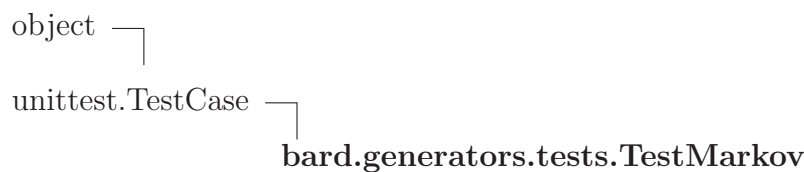
<code>issents(self)</code>

8 Module `bard.generators.tests`

8.1 Variables

Name	Description
<code>__package__</code>	Value: <code>'bard.generators'</code>

8.2 Class `TestMarkov`



Known Subclasses: `bard.generators.tests.TestIntelligentMarkov`, `bard.generators.tests.TestSentenceBase`

8.2.1 Methods

`setUp(self)`

Hook method for setting up the test fixture before exercising it.

Overrides: `unittest.TestCase.setUp` `exitit`(inherited documentation)

`test_generator(self)`

`test_tagged_generator(self)`

Inherited from `unittest.TestCase`

`__call__()`, `__eq__()`, `__hash__()`, `__init__()`, `__ne__()`, `__repr__()`, `__str__()`, `assertAlmostEqual()`, `assertAlmostEquals()`, `assertEqual()`, `assertEquals()`, `assertFalse()`, `assertNotAlmostEqual()`, `assertNotAlmostEquals()`, `assertNotEqual()`, `assertNotEquals()`, `assertRaises()`, `assertTrue()`, `assert_()`, `countTestCases()`, `debug()`, `defaultTestResult()`, `fail()`, `failIf()`, `failIfAlmostEqual()`, `failIfEqual()`, `failUnless()`, `failUnlessAlmostEqual()`, `failUnlessEqual()`, `failUnlessRaises()`, `id()`, `run()`, `shortDescription()`, `tearDown()`

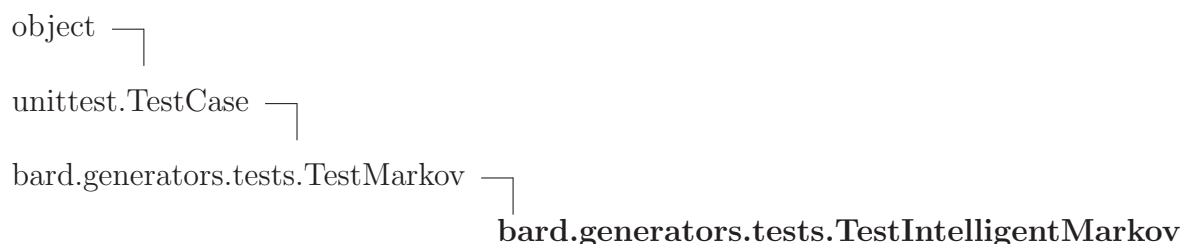
Inherited from `object`

`__delattr__()`, `__format__()`, `__getattr__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__setattr__()`, `__sizeof__()`, `__subclasshook__()`

8.2.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

8.3 Class *TestIntelligentMarkov*



8.3.1 Methods

setUp(*self*)

Hook method for setting up the test fixture before exercising it.

Overrides: `unittest.TestCase.setUp` `exitit` (inherited documentation)

Inherited from `bard.generators.tests.TestMarkov` (Section 8.2)

`test_generator()`, `test_tagged_generator()`

Inherited from `unittest.TestCase`

`__call__()`, `__eq__()`, `__hash__()`, `__init__()`, `__ne__()`, `__repr__()`, `__str__()`, `assertAlmostEqual()`, `assertAlmostEquals()`, `assertEqual()`, `assertEquals()`, `assertFalse()`, `assertNotAlmostEqual()`, `assertNotAlmostEquals()`, `assertNotEqual()`, `assertNotEquals()`, `assertRaises()`, `assertTrue()`, `assert_()`, `countTestCases()`, `debug()`, `defaultTestResult()`, `fail()`, `failIf()`, `failIfAlmostEqual()`, `failIfEqual()`, `failUnless()`, `failUnlessAlmostEqual()`, `failUnlessEqual()`, `failUnlessRaises()`, `id()`, `run()`, `shortDescription()`, `tearDown()`

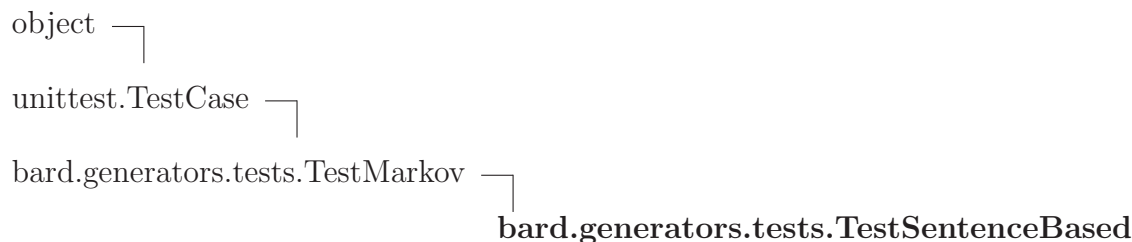
Inherited from `object`

`__delattr__()`, `__format__()`, `__getattr__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__setattr__()`, `__sizeof__()`, `__subclasshook__()`

8.3.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

8.4 Class *TestSentenceBased*



8.4.1 Methods

setUp(*self*)

Hook method for setting up the test fixture before exercising it.

Overrides: `unittest.TestCase.setUp` exitit(inherited documentation)

Inherited from `bard.generators.tests.TestMarkov`(Section 8.2)

`test_generator()`, `test_tagged_generator()`

Inherited from `unittest.TestCase`

`--call--()`, `--eq--()`, `--hash--()`, `--init--()`, `--ne--()`, `--repr--()`, `--str--()`, `assertAlmostE-qual()`, `assertAlmostEquals()`, `assertEqual()`, `assertEquals()`, `assertFalse()`, `assert-NotAlmostEqual()`, `assertNotAlmostEquals()`, `assertNotEqual()`, `assertNotEquals()`, `assertRaises()`, `assertTrue()`, `assert_()`, `countTestCases()`, `debug()`, `defaultTestRe-sult()`, `fail()`, `failIf()`, `failIfAlmostEqual()`, `failIfEqual()`, `failUnless()`, `failUnlessAlmostE-qual()`, `failUnlessEqual()`, `failUnlessRaises()`, `id()`, `run()`, `shortDescription()`, `tear-Down()`

Inherited from `object`

`--delattr--()`, `--format--()`, `--getattrattribute--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`, `--setattr--()`, `--sizeof--()`, `--subclasshook--()`

8.4.2 Properties

Name	Description
<i>Inherited from object</i> __class__	

9 Module bard.test

Index

- bard (*package*), 2
 - bard.detokenizers (*package*), 3
 - bard.detokenizers.detokenize (*function*), 3
 - bard.detokenizers.regex (*module*), 4
 - bard.detokenizers.tests (*module*), 5–6
 - bard.generate_text (*function*), 2
 - bard.generators (*package*), 7
 - bard.generators.generate (*function*), 7
 - bard.generators.markov (*module*), 8–10
 - bard.generators.sentence (*module*), 11
 - bard.generators.tests (*module*), 12–15
 - bard.test (*module*), 16