

Detección de sexismo en Twitter
Ingeniería de Software

Abraham Solís Álvarez
Mario Alejandro Gil Lázaro

December 8, 2016

Contents

Contents	1
1 Requerimientos	3
1.1 Requerimientos	3
1.1.1 Descripción del proyecto	3
1.1.2 Requerimientos del proyecto	3
1.1.3 Dependencias	3
2 Desarrollo	4
2.1 Desarrollo	5
2.1.1 Cronograma de Actividades	5
2.1.2 Reportes de Revisiones técnicas e informales	6
2.1.3 Bugs corregidos	6
3 Documentación	7
3.1 Documentación	7
3.1.1 Cómo se usa el software	7
3.1.2 Documentación de las funciones internas	7
3.2 Package analisis-machismo	8
3.2.1 Modules	8
3.3 Package analisis-machismo.app	9
3.3.1 Modules	9
3.4 Module analisis-machismo.app.analysis	10
3.4.1 Functions	10
3.5 Module analisis-machismo.app.dictionary_tagger	11
3.5.1 Class DictionaryTagger	11
3.6 Module analisis-machismo.app.key_reader	12
3.6.1 Class KeyReader	12
3.7 Module analisis-machismo.app.tag_counter	13
3.7.1 Class TagCounter	13
3.8 Module analisis-machismo.app.tweet_formatter	14
3.8.1 Class TweetFormatter	14
3.9 Module analisis-machismo.app.twitter_miner	15
3.9.1 Class TwitterMiner	15
3.10 Package analisis-machismo.tests	16
3.10.1 Modules	16
3.11 Module analisis-machismo.tests.test_dictionary_tagger	17
3.11.1 Class TestDictionaryTagger	17
3.12 Module analisis-machismo.tests.test_key_reader	18
3.12.1 Class TestKeyReader	18

3.13	Module analisis-machismo.tests.test_tweet_formatter	19
3.13.1	Class TestTweetFormatter	19
4	Resultado	20
4.1	Resultado	20
4.1.1	Modulos en /app	20
4.1.2	Modulos en /test	24
5	Conclusiones	29
5.1	Conclusiones	29
	Index	30

Chapter 1

Requerimientos

1.1 Requerimientos

1.1.1 Descripción del proyecto

Este proyecto busca determinar los niveles de sexismo y otras manifestaciones de odio y discriminación, en el texto que los usuarios del sitio de microblogueo Twitter, escriben diariamente. Dado que el internet es una plataforma moderna de expresión y debido también a que la ya mencionada red social posee un número importante de usuarios, consideramos que la información ahí recabada representa una muestra importante. Así mismo, el hecho de que los medios virtuales son comúnmente considerados como medios poco trascendentales, en los que se puede supuestamente evitar las consecuencias que los propios comentarios puedan ocasionar, creemos que las opiniones ahí expresadas poseen un alto grado de sinceridad, mayor al que podría obtenerse en el discurso hablado.

1.1.2 Requerimientos del proyecto

El sistema propuesto debe ser capaz de recolectar los posts directamente del stream de Twitter, convertir los datos al formato que mejor convenga, etiquetar palabra por palabra el texto, y realizar mediciones que permitan elaborar indicadores que arrojen luz sobre las costumbres y la cultura de los usuarios de la red social. Se pretende lograr esto manteniendo los más altos estándares de calidad que sea posible, aplicando prácticas de programación modernas.

1.1.3 Dependencias

Para ejecutar el programa con las mayores probabilidades de éxito se recomienda utilizar un sistema operativo basado en GNU/Linux. Es necesario instalar una Python 3.x, de preferencia la versión 3.5, que fue la utilizada en el desarrollo del sistema. Se recomienda utilizar el programa ‘pip’ para instalar los módulos de Python ‘nltk’ y ‘plac’, que son necesarios para ejecutar el programa. Es necesario contar con una conexión a internet.

Chapter 2

Desarrollo

2.1 Desarrollo

2.1.1 Cronograma de Actividades

Cronograma de actividades

De acuerdo con las etapas que se deben llevar a cabo para la realización del proyecto, se presenta el siguiente cronograma de commits.

Commit \ Días	Sat Dec 3	Sun Dec 4	Mon Dec 5	Tue Dec 6	Wed Dec 7	Sat Dec 8
Initial commit						
Read keys from .ini file						
Add machist dictionary						
Add readkeys unit tests						
Separate app and tests code						
Data belongs to app folder						
Write script to automate tests						
Read tracks file						
Add filter tracks file						
Update local repository						
Change to read actual track file						
Remove quotes from tracks file						
Update visualizer to show only the text of the tweet						
Add KeyboardInterrupt Exception						
Add filter tracks file						
Update machist dictionary						
Sexist tagger ready to be plugged						
Update local repository						
Update .gitignore file						
Bring spaghetti files to use trained taggers						
Resolve .gitignore conflict						
Make key_reader object oriented						
Use more concise names						
Pack spaghetti module						
Refactor twitter_streaming module to make TwitterMiner class						
Reduce console output when keyboard interrupt						
Refactor visualizer.py code to make it OO						

Add regex to clean tweets in
tweet_formatter
Add tests for tweet_formatter.py and
make it pass
Test compile dictionaries from
dictionary_tagger
Test tag method in dictionary_tagger
Add 'stop after # tweets retrieved'
functionality to listener
Write some docstrings
Add some docstring
Split dictionaries
Changed dictionaries
Repaired tabs inconsistencies
Add some docstring
Use plac to generate -h menu for
analysis.py
Update local repository
Add some docstring
Add a tag counter
Update local repository
Add a usage prompt to main function
with plac
Improve analysis documentation
Review key_reader documentation
Rewrite tag_counter documentation
Update tweet_formatter documentation
Update twitter_miner documentation



2.1.2 Reportes de Revisiones técnicas e informales

La información de los reportes y bugs que se fueron corrigiendo aparecen en los commits hechos en la url del proyecto. <https://github.com/azaroma/analisis-machismo>

2.1.3 Bugs corregidos

La información de los reportes y bugs que se fueron corrigiendo aparecen en los commits hechos en la url del proyecto. <https://github.com/azaroma/analisis-machismo>

Chapter 3

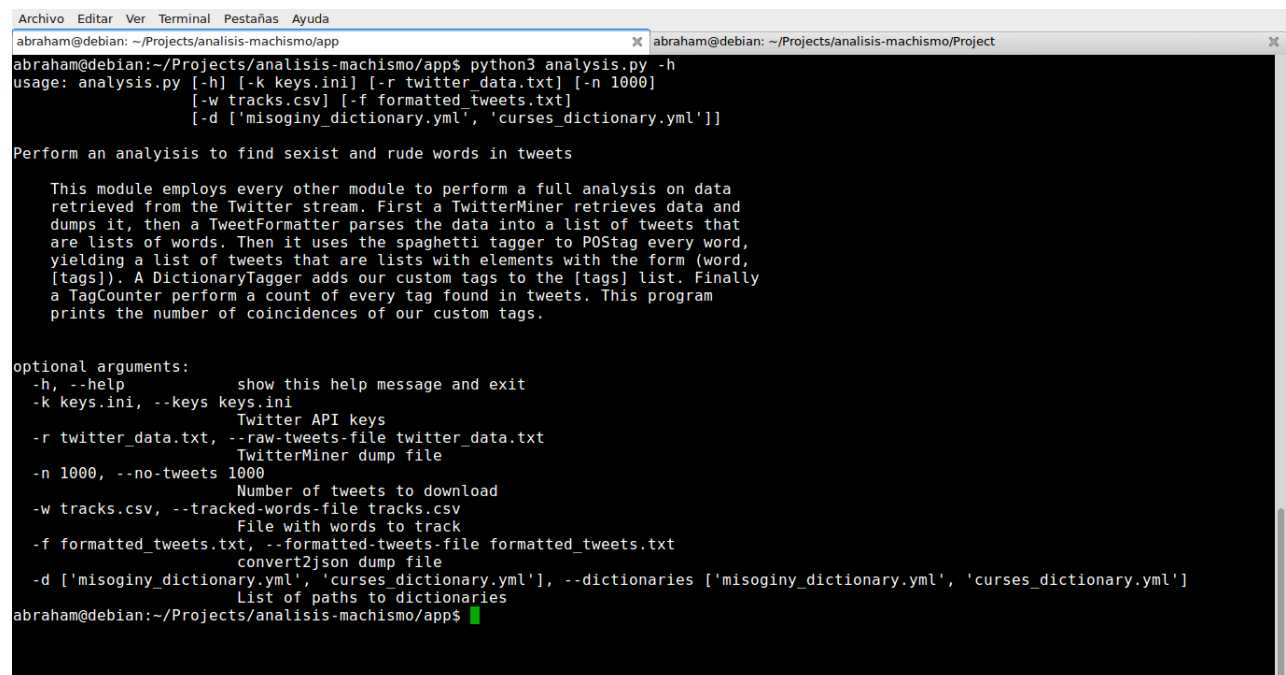
Documentación

3.1 Documentación

3.1.1 Cómo se usa el software

Por el momento, la única forma de interactuar con el sistema es mediante el módulo ‘analysis.py’, localizado en el directorio ‘app’ (con el comando ‘python analysis.py’). Tenga en cuenta que usted debe estar localizado dentro de dicho directorio. Además, se proporciona un script que ejecuta las pruebas del sistema en el directorio raíz: ‘run_tests’.

3.1.2 Documentación de las funciones internas



```
Archivo Editar Ver Terminal Pestañas Ayuda
abraham@debian: ~/Projects/analisis-machismo/app
abraham@debian:~/Projects/analisis-machismo/app$ python3 analysis.py -h
usage: analysis.py [-h] [-k keys.ini] [-r twitter_data.txt] [-n 1000]
                  [-w tracks.csv] [-f formatted_tweets.txt]
                  [-d ['misoginy_dictionary.yml', 'curses_dictionary.yml']]

Perform an analysis to find sexist and rude words in tweets

This module employs every other module to perform a full analysis on data
retrieved from the Twitter stream. First a TwitterMiner retrieves data and
dumps it, then a TweetFormatter parses the data into a list of tweets that
are lists of words. Then it uses the spaghetti tagger to POSTag every word,
yielding a list of tweets that are lists with elements with the form (word,
[tags]). A DictionaryTagger adds our custom tags to the [tags] list. Finally
a TagCounter perform a count of every tag found in tweets. This program
prints the number of coincidences of our custom tags.

optional arguments:
  -h, --help            show this help message and exit
  -k keys.ini, --keys keys.ini
                        Twitter API keys
  -r twitter_data.txt, --raw-tweets-file twitter_data.txt
                        TwitterMiner dump file
  -n 1000, --no-tweets 1000
                        Number of tweets to download
  -w tracks.csv, --tracked-words-file tracks.csv
                        File with words to track
  -f formatted_tweets.txt, --formatted-tweets-file formatted_tweets.txt
                        convert2json dump file
  -d ['misoginy_dictionary.yml', 'curses_dictionary.yml'], --dictionaries ['misoginy_dictionary.yml', 'curses_dictionary.yml']
                        List of paths to dictionaries
abraham@debian:~/Projects/analisis-machismo/app$
```

3.2 Package analisis-machismo

3.2.1 Modules

- **app** (*Section 3.3, p. 9*)
 - **analysis** (*Section 3.4, p. 10*)
 - **dictionary_tagger** (*Section 3.5, p. 11*)
 - **key_reader** (*Section 3.6, p. 12*)
 - **tag_counter** (*Section 3.7, p. 13*)
 - **tweet_formatter** (*Section 3.8, p. 14*)
 - **twitter_miner** (*Section 3.9, p. 15*)
- **tests** (*Section 3.10, p. 16*)
 - **test_dictionary_tagger** (*Section 3.11, p. 17*)
 - **test_key_reader** (*Section 3.12, p. 18*)
 - **test_tweet_formatter** (*Section 3.13, p. 19*)

3.3 Package analisis-machismo.app

3.3.1 Modules

- **analysis** (*Section 3.4, p. 10*)
- **dictionary_tagger** (*Section 3.5, p. 11*)
- **key_reader** (*Section 3.6, p. 12*)
- **tag_counter** (*Section 3.7, p. 13*)
- **tweet_formatter** (*Section 3.8, p. 14*)
- **twitter_miner** (*Section 3.9, p. 15*)

3.4 Module analisis-machismo.app.analysis

3.4.1 Functions

```
main(keys='keys.ini', raw_tweets_file='twitter_data.txt', no_tweets=1000,  
tracked_words_file='tracks.csv', formatted_tweets_file='formatted_tweets.txt',  
dictionaries=['misoginy_dictionary.yml', 'curses_dictionary.yml'])
```

Perform an analysis to find sexist and rude words in tweets

This module employs every other module to perform a full analysis on data retrieved from the Twitter stream. First a TwitterMiner retrieves data and dumps it, then a TweetFormatter parses the data into a list of tweets that are lists of words. Then it uses the spaghetti tagger to POSTag every word, yielding a list of tweets that are lists with elements with the form (word, [tags]). A DictionaryTagger adds our custom tags to the [tags] list. Finally a TagCounter perform a count of every tag found in tweets. This program prints the number of coincidences of our custom tags.

3.5 Module `analisiis-machismo.app.dictionary_tagger`

3.5.1 Class `DictionaryTagger`

Python class for tagging text with dictionaries

Methods

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

Dictionary is a dict containing all the dictionaries parsed from the paths given.

<code>compile_dictionaries(self, dictionary_paths)</code>
--

Returns a list of dictionaries parsed from .yaml files
--

<code>tag(self, postagged_sentences)</code>
--

<code>tag_sentence(self, sentence, tag_with_lemmas=None)</code>
--

The result is only tagging of all the possible ones. The resulting tagging is determined by these two priority rules:

- | |
|---|
| <ul style="list-style-type: none">• longest matches have higher priority• search is made left to right |
|---|

3.6 Module analisis-machismo.app.key_reader

3.6.1 Class KeyReader

Wrapper of ConfigParser to load .ini file with Twitter API keys

Methods

<code>__init__(self)</code>

<code>read(self, filename='keys.ini', section='keys')</code>
--

Import the configparser, tell it to read the file, and get a listing of the sections. Sections are listed in a python dictionary.

3.7 Module *alisis-machismo.app.tag_counter*

3.7.1 Class `TagCounter`

Wrapper for a tag counting method

Methods

<code>__init__</code> (<i>self</i> , <i>tagged_tweets</i>)
Store a list of tweets in case none is provided in <code>count()</code>

<code>count</code> (<i>self</i> , <i>tagged_tweets</i> =None)
Handle the counting of tags inserting them in a dictionary to ensure their uniqueness. Can manage a list of tags and a single string tag.

3.8 Module *alisis-machismo.app.tweet_formatter*

3.8.1 Class `TweetFormatter`

Provides the tools needed to format raw data from Twitter stream to stripped text like this: raw -> json -> text -> stripped text.

Methods

<code>__init__</code> (<i>self</i> , <i>source_file</i> =None)
--

<code>convert2json</code> (<i>self</i> , <i>tweets_source</i> =None)
--

Wrap json module to convert raw Twitter data to json
--

<code>convert2text</code> (<i>self</i> , <i>tweets_data</i> =None, <i>output_file</i> =None)
--

Grab 'text' field of jsons only and return a list of them

<code>clean_tweets</code> (<i>self</i> , <i>tweets_text</i> =None)
--

Regular expressions to remove unnecessary characters in Tweets
--

3.9 Module *analisi-machismo.app.twitter_miner*

3.9.1 Class `TwitterMiner`

Exposes the whole chain needed to retrieve data from the Twitter stream from reading the keys, create an oAuth object,

Methods

<code>__init__</code> (<i>self</i> , <i>keys_file</i> ='keys.ini', <i>output_file</i> =None, <i>max_tweets</i> =10)

<code>connect</code> (<i>self</i>)

Get keys and connect to Twitter through OAuth

<code>mine</code> (<i>self</i> , <i>track_words</i> , <i>output_file</i> =None)

Retrieve tweets with text that matches track_words.

3.10 Package analisis-machismo.tests

3.10.1 Modules

- `test_dictionary_tagger` (*Section 3.11, p. 17*)
- `test_key_reader` (*Section 3.12, p. 18*)
- `test_tweet_formatter` (*Section 3.13, p. 19*)

3.11 Module `analysis-machismo.tests.test_dictionary_tagger`

3.11.1 Class `TestDictionaryTagger`

`unittest.TestCase` — `analysis-machismo.tests.test_dictionary_tagger.TestDictionaryTagger`

Methods

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

<p><code>unittest</code> supports test automation, sharing of setup and shutdown code for tests, aggregation of tests into collections, and independence of the tests from the reporting framework. The <code>unittest</code> module provides classes that make it easy to support these qualities for a set of tests.</p>
--

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

3.12 Module `analysis-machismo.tests.test_key_reader`

3.12.1 Class `TestKeyReader`

`unittest.TestCase` —
`analysis-machismo.tests.test_key_reader.TestKeyReader`

Methods

<code>setup(self)</code>

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

unittest supports test automation, sharing of setup and shutdown code for tests, aggregation of tests into collections, and independence of the tests from the reporting framework. The unittest module provides classes that make it easy to support these qualities for a set of tests.

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

3.13 Module `analysis-machismo.tests.test_tweet_formatter`

3.13.1 Class `TestTweetFormatter`

`unittest.TestCase` └─
 `analysis-machismo.tests.test_tweet_formatter.TestTweetFormatter`

The crux of each test is a call to `assertEqual()` to check for an expected result; `assertTrue()` or `assertFalse()` to verify a condition; or `assertRaises()` to verify that a specific exception gets raised. These methods are used instead of the `assert` statement so the test runner can accumulate all test results and produce a report.

Methods

`test_inexistent_file(self)`

`test_no_JSON_tweets(self)`

`test_no_text_tweets(self)`

`test_no_text_key(self)`

Chapter 4

Resultado

4.1 Resultado

4.1.1 Modulos en /app

Segun el analisis estático del código fuente en Python hecho por Pylint hecho a los modulos localizados en */app*.

```
***** Module analysis
W: 54, 0: Cannot decode using encoding "ascii", unexpected byte at position 27
(invalid-encoded-data)
W: 59, 0: Cannot decode using encoding "ascii", unexpected byte at position 56
(invalid-encoded-data)
C: 33, 0: Trailing whitespace (trailing-whitespace)
C: 47, 0: Trailing whitespace (trailing-whitespace)
C: 54, 0: Unnecessary parens after 'print' keyword (superfluous-parens)
C: 59, 0: Unnecessary parens after 'print' keyword (superfluous-parens)
C: 1, 0: Missing module docstring (missing-docstring)
F: 1, 0: Unable to import 'plac' (import-error)
F: 3, 0: Unable to import 'spaghetti' (import-error)
W: 20, 0: Dangerous default value ['misogyny_dictionary.yml', 'curses_dictionary.yml']
as argument (dangerous-default-value)
R: 20, 0: Too many arguments (6/5) (too-many-arguments)
***** Module dictionary_tagger
C: 31, 0: Trailing whitespace (trailing-whitespace)
C: 1, 0: Missing module docstring (missing-docstring)
C: 3, 0: Old-style class defined. (old-style-class)
W: 25,12: Redefining built-in 'file' (redefined-builtin)
R: 20, 4: Method could be a function (no-self-use)
C: 29, 4: Missing method docstring (missing-docstring)
W: 32,37: Unused argument 'tag_with_lemmas' (unused-argument)
***** Module __init__
C: 1, 0: Missing module docstring (missing-docstring)
***** Module key_reader
C: 9, 0: Trailing whitespace (trailing-whitespace)
```

```

C: 11, 0: Trailing whitespace (trailing-whitespace)
C: 14, 0: Trailing whitespace (trailing-whitespace)
C: 23, 0: Line too long (82/80) (line-too-long)
C: 1, 0: Missing module docstring (missing-docstring)
C: 3, 0: Old-style class defined. (old-style-class)
R: 3, 0: Too few public methods (1/2) (too-few-public-methods)
***** Module tag_counter
C: 3, 0: Trailing whitespace (trailing-whitespace)
C: 6, 0: Trailing whitespace (trailing-whitespace)
C: 11, 0: Trailing whitespace (trailing-whitespace)
C: 33, 0: Trailing whitespace (trailing-whitespace)
C: 1, 0: Missing module docstring (missing-docstring)
C: 1, 0: Old-style class defined. (old-style-class)
R: 1, 0: Too few public methods (1/2) (too-few-public-methods)
***** Module tweet_formatter
C: 6, 0: Trailing whitespace (trailing-whitespace)
C: 15, 0: Trailing whitespace (trailing-whitespace)
C: 56, 0: Line too long (131/80) (line-too-long)
C: 1, 0: Missing module docstring (missing-docstring)
C: 5, 0: Old-style class defined. (old-style-class)
W: 28,16: No exception type(s) specified (bare-except)
R: 32, 4: Method could be a function (no-self-use)
R: 49, 4: Method could be a function (no-self-use)
W: 1, 0: Unused import sys (unused-import)
***** Module twitter_miner
C: 23, 0: Trailing whitespace (trailing-whitespace)
C: 23, 0: Trailing whitespace (trailing-whitespace)
C: 26, 0: Unnecessary parens after 'print' keyword (superfluous-parens)
C: 34, 0: Unnecessary parens after 'print' keyword (superfluous-parens)
C: 40, 0: Unnecessary parens after 'print' keyword (superfluous-parens)
C: 44, 0: Trailing whitespace (trailing-whitespace)
C: 58, 0: Line too long (84/80) (line-too-long)
C: 59, 0: Line too long (90/80) (line-too-long)
C: 68, 0: Trailing whitespace (trailing-whitespace)
C: 74, 0: Unnecessary parens after 'print' keyword (superfluous-parens)
C: 77, 0: Unnecessary parens after 'print' keyword (superfluous-parens)
C: 1, 0: Missing module docstring (missing-docstring)
W: 17,28: Unused variable 'err' (unused-variable)
C: 42, 0: Old-style class defined. (old-style-class)
C: 69,39: Invalid variable name "t" (invalid-name)
W: 60, 8: Attribute 'stream' defined outside __init__ (attribute-defined-outside-init)
W: 55, 8: Attribute 'keys' defined outside __init__ (attribute-defined-outside-init)
W: 54, 8: Attribute 'key_reader' defined outside __init__ (attribute-defined-outside-init)

```

Report

189 statements analysed.

Raw metrics

+-----+-----+-----+-----+-----+

type	number	%	previous	difference
code	216	75.52	216	=
docstring	51	17.83	51	=
comment	0	0.00	0	=
empty	19	6.64	19	=

External dependencies

```
::
```

```

configparser (key_reader)
dictionary_tagger (analysis)
key_reader (twitter_miner)
tag_counter (analysis)
tweepy (twitter_miner)
    \-streaming (twitter_miner)
tweet_formatter (analysis)
yaml (dictionary_tagger)

```

Duplication

	now	previous	difference
nb duplicated lines	0	0	=
percent duplicated lines	0.000	0.000	=

Messages by category

type	number	previous	difference
convention	41	41	=
refactor	6	6	=
warning	11	11	=
error	0	0	=

% errors / warnings by module

module	error	warning	refactor	convention
twitter_miner	0.00	36.36	0.00	34.15
analysis	0.00	27.27	16.67	12.20
tweet_formatter	0.00	18.18	33.33	12.20
dictionary_tagger	0.00	18.18	16.67	9.76

Messages

message id	occurrences
trailing-whitespace	16
missing-docstring	8
superfluous-parens	7
old-style-class	5
line-too-long	4
no-self-use	3
attribute-defined-outside-init	3
too-few-public-methods	2
invalid-encoded-data	2
import-error	2
unused-variable	1
unused-import	1
unused-argument	1
too-many-arguments	1
redefined-builtin	1
invalid-name	1
dangerous-default-value	1

bare-except	1	
-------------	---	--

Global evaluation

Your code has been rated at 6.93/10 (previous run: 6.93/10, +0.00)

Statistics by type

type	number	old number	difference	%documented	%badname	
module	7	7	=	0.00	0.00	
class	6	6	=	100.00	0.00	
method	18	18	=	94.44	0.00	
function	1	1	=	100.00	0.00	

4.1.2 Modulos en /test

Segun el analisis estático del código fuente en Python hecho por Pylint hecho a los modulos localizados en /test.

```

***** Module __init__
C:  1, 0: Missing module docstring (missing-docstring)
***** Module test_dictionary_tagger
W: 16, 0: Cannot decode using encoding "ascii", unexpected byte at position 16
(invalid-encoded-data)
W: 41, 0: Cannot decode using encoding "ascii", unexpected byte at position 91
(invalid-encoded-data)
W: 42, 0: Cannot decode using encoding "ascii", unexpected byte at position 96
(invalid-encoded-data)
C:  7, 0: Trailing whitespace (trailing-whitespace)
C:  8, 0: Trailing whitespace (trailing-whitespace)
C:  9, 0: Trailing whitespace (trailing-whitespace)
C:  9, 0: Line too long (81/80) (line-too-long)
C: 10, 0: Trailing whitespace (trailing-whitespace)
C: 10, 0: Line too long (81/80) (line-too-long)
C: 31, 0: Wrong continued indentation.
                                compile_dictionaries(['tests/subset_dict.yml']),
                                |      ^ (bad-continuation)
C: 32, 0: Wrong continued indentation.
                                [d])
                                |      ^ (bad-continuation)
C: 34, 0: Wrong continued indentation.
                                compile_dictionaries(['tests/subset_dict.yml']

```



```

|      ^ (bad-continuation)
C: 35, 0: Exactly one space required after comma
                                     , 'tests/subset2_dict.yml'] ) ,
                                     ^ (bad-whitespace)
C: 36, 0: Trailing whitespace (trailing-whitespace)
C: 36, 0: Exactly one space required after comma
                                     [d,d2])
                                     ^ (bad-whitespace)
C: 41, 0: Line too long (214/80) (line-too-long)
C: 42, 0: Line too long (227/80) (line-too-long)
C: 1, 0: Missing module docstring (missing-docstring)
F: 2, 0: Unable to import 'app.dictionary_tagger' (import-error)
C: 4, 0: Missing class docstring (missing-docstring)
C: 14, 8: Invalid variable name "d" (invalid-name)
C: 21, 8: Invalid variable name "d2" (invalid-name)
C: 38, 4: Missing method docstring (missing-docstring)
R: 4, 0: Too many public methods (47/20) (too-many-public-methods)
***** Module test_key_reader
C: 12, 0: Line too long (81/80) (line-too-long)
C: 13, 0: Line too long (81/80) (line-too-long)
C: 1, 0: Missing module docstring (missing-docstring)
F: 2, 0: Unable to import 'app.key_reader' (import-error)
C: 4, 0: Missing class docstring (missing-docstring)
C: 6, 4: Missing method docstring (missing-docstring)
C: 16, 8: Invalid variable name "kr" (invalid-name)
C: 19, 4: Missing method docstring (missing-docstring)
C: 26, 8: Invalid variable name "kr" (invalid-name)
R: 4, 0: Too many public methods (48/20) (too-many-public-methods)
***** Module test_tweet_formatter
C: 6, 0: Trailing whitespace (trailing-whitespace)
C: 7, 0: Trailing whitespace (trailing-whitespace)
C: 8, 0: Trailing whitespace (trailing-whitespace)
C: 1, 0: Missing module docstring (missing-docstring)
F: 2, 0: Unable to import 'app.tweet_formatter' (import-error)
C: 11, 4: Missing method docstring (missing-docstring)
C: 12, 8: Invalid variable name "f" (invalid-name)
C: 13, 8: Invalid variable name "f2" (invalid-name)
E: 14,26: Undefined variable 'FileNotFoundError' (undefined-variable)
E: 16,26: Undefined variable 'FileNotFoundError' (undefined-variable)
E: 17,26: Undefined variable 'FileNotFoundError' (undefined-variable)
C: 19, 4: Invalid method name "test_no_JSON_tweets" (invalid-name)
C: 19, 4: Missing method docstring (missing-docstring)
C: 20, 8: Invalid variable name "f" (invalid-name)
C: 24, 8: Invalid variable name "f2" (invalid-name)
W: 24, 8: Unused variable 'f2' (unused-variable)
C: 29, 4: Missing method docstring (missing-docstring)
C: 30, 8: Invalid variable name "f" (invalid-name)
C: 34, 8: Invalid variable name "f2" (invalid-name)
W: 34, 8: Unused variable 'f2' (unused-variable)
C: 38, 4: Missing method docstring (missing-docstring)
C: 39, 8: Invalid variable name "f" (invalid-name)
R: 4, 0: Too many public methods (49/20) (too-many-public-methods)

```

Report

63 statements analysed.

Messages by category

type	number	previous	difference
convention	44	44	=
refactor	3	3	=
warning	5	5	=
error	3	3	=

% errors / warnings by module

module	error	warning	refactor	convention
test_tweet_formatter	100.00	40.00	33.33	36.36
test_dictionary_tagger	0.00	60.00	33.33	43.18

Messages

message id	occurrences
missing-docstring	13
invalid-name	12
trailing-whitespace	8
line-too-long	6
undefined-variable	3

too-many-public-methods	3	
invalid-encoded-data	3	
import-error	3	
bad-continuation	3	
unused-variable	2	
bad-whitespace	2	

Global evaluation

Your code has been rated at $-0.63/10$ (previous run: $-0.63/10$, $+0.00$)

Duplication

	now	previous	difference
nb duplicated lines	0	0	=
percent duplicated lines	0.000	0.000	=

Raw metrics

type	number	%	previous	difference
code	71	63.39	71	=
docstring	32	28.57	32	=
comment	0	0.00	0	=
empty	9	8.04	9	=

Statistics by type

type	number	old number	difference	%documented	%badname
module	4	4	=	0.00	0.00
class	3	3	=	33.33	0.00
method	9	9	=	22.22	11.11
function	0	0	=	0	0

Chapter 5

Conclusiones

5.1 Conclusiones

El machismo ha sido uno de los problemas más importantes de la sociedad en todas las épocas y en todos los sentidos.

El machismo ha ayudado negativamente a la no liberación de la mujer, muchas de ellas han vivido a la sombra de unos hombres que han pensado equivocadamente que son seres superiores, otras en cambio han sido fuertes y han luchado contra este gran problema.

El problema que ha tenido el machismo es que los pensadores de las épocas determinadas apoyaban el machismo y afirmaban una inferioridad mental y física de la mujer, en algunas épocas los pensadores estaban muy seguidos por la sociedad y la sociedad pensaba igual que lo que pensaban los pensadores más famosos.

En las diferentes épocas tienen problemas muy diversos, pero todos llegan al mismo punto, que la mujer no debe escribir y si escribe sus obras tienen menos importancia y menos calidad que la de los hombres. Estas escritoras tenían dificultades incluso para tener un estudio digno y muchas de ellas intentaban hacerse pasar por hombres para que sus obras tuvieran éxito.

Lo que si es evidente es que muchas mujeres que han tenido un gran potencial psicológico y que por culpa de este problema no han podido dar a conocer todas sus posibilidades.

Es importante el papel que han hecho aquellas mujeres valientes que han luchado contra el machismo y han expresado mediante su gran virtud, que son los libros, expresar y sacar a la luz todos los problemas que han tenido y como han expresado su ayuda a todas aquellas mujeres que han vivido y viven escondidas por una idealización equivocada.

No se ha tratado igual de mal a todas las autoras, muchas de ellas eran respetadas por el tema religioso, había muchas autoras que se consideraban la fuente de expresión divina y por esos sus pensamientos eran escuchados y respetados.

Actualmente el tema del machismo tiene un grado de importancia menor y en estos tiempos hay muchas más autoras que en épocas anteriores y sus obras son igualmente respetadas y admiradas como la de cualquier hombre.

Incluso en el siglo XX la literatura feminista, ignorada en épocas anteriores, tiene un papel muy importante en nuestra sociedad.

Index

- analysis-machismo (*package*), 2
 - analysis-machismo.app (*package*), 3
 - analysis-machismo.app.analysis (*module*), 4
 - analysis-machismo.app.dictionary_tagger (*module*), 5
 - analysis-machismo.app.key_reader (*module*), 6
 - analysis-machismo.app.tag_counter (*module*), 7
 - analysis-machismo.app.tweet_formatter (*module*), 8
 - analysis-machismo.app.twitter_miner (*module*), 9
 - analysis-machismo.tests (*package*), 10
 - analysis-machismo.tests.test_dictionary_tagger (*module*), 11
 - analysis-machismo.tests.test_key_reader (*module*), 12
 - analysis-machismo.tests.test_tweet_formatter (*module*), 13