**Sentiments**

[**tl;dr**](https://docs.cs50.net/problems/sentiments/sentiments.html#tl-dr)

1. Implemente un programa que categorice una palabra como positiva o negativa.
2. $ ./smile love
3. :)
4. $ ./smile hate
5. :(
6. $ ./smile Stanford

:|

1. Implemente un programa que categorices los tweets de un usuario como positivos o negativos.
2. $ ./tweets @cs50
3. 0 hello, @world
4. 1 I love you, @world
5. -1 I hate you, @world

...

1. Implemente un website que genere a pie chart categorizing a user’s tweets.

[**Background**](https://docs.cs50.net/problems/sentiments/sentiments.html#background)

"Sentiment analysis," también conocido como "opinion mining," involucra inferencia de sentimientos (i.e., opinión) desde un texto. Por ejemplo, las reseñas de película de [Rotten Tomatoes](https://www.rottentomatoes.com/) son a menudo positivas o negativas. También lo son las reseñas de productos en [Amazon](https://www.amazon.com/). Del mismo modo, las opiniones que subyacen en muchos tweets en [Twitter](https://twitter.com/)..

Algunas palabras tienden a tener connotaciones positivas (ej. "love" o amor), mientras que algunas palabras tienden a tener connotaciones negativas (ej., "hate" u odio). Y así, si alguien fuese a twittear "I love you" (te amo), usted probablemente infiera un sentimiento positivo. Y si alguien fuese a twittear "I hate you"(te odio), podría inferir un sentimiento negativo. Por supuesto, las palabras individuales por sí solas no siempre son confiables, ya que “I don’t love you (no te amo) probablemente no sea un sentimiento positivo, pero no nos preocupemos por esos casos. Algunas palabras, mientras tanto, no tienen connotaciones positivas ni negativas

Hace algunos años atrás, el Dr. Minqing Hu y el Prof. Bing Liu de la University of Illinois en Chicago gentilmente juntaron una [lista](https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#lexicon) de [2006 palabras positivas](https://raw.githubusercontent.com/cs50/problems/sentiments/positive-words.txt) y [4783 palabras negativas](https://raw.githubusercontent.com/cs50/problems/sentiments/negative-words.txt). Nosotros la utilizaremos para clasificar los tweets! Pero antes un tour.

[**Distribución**](https://docs.cs50.net/problems/sentiments/sentiments.html#distribution)

[**Descarga**](https://docs.cs50.net/problems/sentiments/sentiments.html#downloading)

$ wget <https://github.com/cs50/problems/archive/sentiments.zip>

$ unzip sentiments.zip

$ rm sentiments.zip

$ mv problems-sentiments sentiments

$ cd sentiments

$ chmod a+x smile tweets

$ ls

analyzer.py helpers.py positive-words.txt smile\* tweets\*

application.py negative-words.txt requirements.txt templates/

[**Entendiéndo**](https://docs.cs50.net/problems/sentiments/sentiments.html#understanding)

[**smile**](https://docs.cs50.net/problems/sentiments/sentiments.html#code-smile-code)

Abra smile en sentiments/. Basta decir que el nombre del archivo no termina en .py, ¡A pesar de que el archivo contiene un programa escrito en Python, pero eso está bien! Observe el asunto encima del archivo:

#!/usr/bin/env python3

Esa línea le dice a la computadora que interprete (corra) el programa usando python3 (alias python en CS50 IDE), un intérprete que entiende Python 3.

Observe a continuación que el programa importa una clase llamada Analyzer de un módulo llamado analyzer así como una función llamada colored de un módulo llamado termcolor. Lo primero que pronto implementará en un archivo llamado analyzer.py. (Recuerde que una clase en Python es como una struct en C excepto que una clase puede también contener funciones, de otra manera conocidas como "métodos" cuando están dentro de una clase.) Este último colorea las salidas en ventanas de terminal, como veremos pronto.

Este programa define solamente una función, main, que es llamada por la última línea del archivo. Dentro de main, primero nos aseguramos de que sys.argv contiene el número esperado de argumentos de línea de comando. Entonces "instanciamos" (alocamos) un bobjeto Analyzer. Luego pasamos a ese objeto el método analyze la palabra que el usuario ha proveído en sys.argv[1]. A como veremos pronto, ese metodo devolverá un int positivo si su entrada es positiva, un int negativo si su entrada es negativa, y 0 si su entrada no es ni negativa ni positiva. El programa por último imprime un smiley a colores consecuentemente.

[**analyzer.py**](https://docs.cs50.net/problems/sentiments/sentiments.html#code-analyzer-py-code)

Abrir anlyzer.py en sentiments/. No mucho ha sucedido ahí dentro (aún)! Note, sin embargo, que este importa la [Natural Language Toolkit](http://www.nltk.org/), entre cuyas características tiene un [tokenizer](http://www.nltk.org/api/nltk.tokenize.html) que usted puede utilizar para dividir un tweet (el cuál es máximamente un objeto str de 140 caracteres) en una list de palabras (pequeños objetos str).

Ahí dentro, está nuestra definición de esa clase Analyzer, la que tiene dos métodos: init, que es llamado cada vez que Analyzer es instanciado; y analyze, que puede ser llamado para analizar algún text. Ese primer método toma dos argumentos en adición a self: positives, cuyo valor es la ruta a un archive de texto que contiene palabras positivas; y negatives, cuyo valor es la ruta a un archivo de texto que contiene palabras negativas. Mientras tanto, analyze toma un argumento en adición a self: a str para ser analizado por sentiment. A través de esa función es (temporalmente) empotrada para retornar 0 sin importar nada más.

Recuerde que los métodos se pasan automáticamente a la primera referencia a self para que tengan una forma de referirse a las "variables de instancia" de los objetos.

[**positive-words.txt, negative-words.txt**](https://docs.cs50.net/problems/sentiments/sentiments.html#code-positive-words-txt-code-code-negative-words-txt-code)

Abra positive-words.txt y negative-words.txt (sin cambiarlas). Tenga en cuenta que encima de cada archivo hay un montón de comentarios, cada uno de los cuales comienza con un; (Sin embargo, esos son solo archivos de texto, por lo que la elección de los autores es arbitraria). Las listas de palabras positivas y negativas, respectivamente, comienzan debajo de esos comentarios, después de una línea en blanco.

[**tweets**](https://docs.cs50.net/problems/sentiments/sentiments.html#code-tweets-code)

Abra tweets. Ah, otro asunto. ¡Pero nada más que un TODO! Mas de esto muy pronto.

[**helpers.py**](https://docs.cs50.net/problems/sentiments/sentiments.html#code-helpers-py-code)

Abre helpers.py. Debería ver dos funciones: chart y get\_user\_timeline. Dado tres valores (positivo, negativo y neutral, cada uno int o un flotante), el gráfico genera HTML (como str) para un gráfico circular que representa esos valores. Dado un nombre de pantalla, mientras tanto, get\_user\_timeline devuelve una lista de tweets (cada uno como str). Esa función usa [Twython](https://twython.readthedocs.io/) (har har), una biblioteca para Python, para recuperar esos tweets a través de [Twitter’s API](https://dev.twitter.com/overview/api) (application programming interface), un servicio gratuito que puede ser consultado programáticamente para tweets. Observe cómo la función espera que existan dos "variables de entorno". Existen variables de entorno dentro de su ventana de terminal,pares key/value que los programas (como los tweets) pueden acceder mediante programación. Pronto utilizaremos dos, API\_KEY y API\_SECRET, para almacenar las credenciales de Twitter.

[**application.py**](https://docs.cs50.net/problems/sentiments/sentiments.html#code-application-py-code)

Abra application.py. En este archivo hay un "controlador" para una aplicación web basada en Flask con dos puntos finales:/ and /search. El primero despliega la más simple de las formas en las que se puede buscar a un usuario de Twitter por su screen name. El segundo muestra uno de esos gráficos circulares que categorizan los tweets de ese usuario. Sin embargo, tenga en cuenta que se supone que el 100% de esos tweets son neutrales.

[**templates/index.html**](https://docs.cs50.net/problems/sentiments/sentiments.html#code-templates-index-html-code)

Abra templates/index.html. Dentro de ahí está la más simple de las formas. Observe cómo averigua a través de url\_for, una función que viene con Flask, a qué URL debe enviarse el formulario

[**templates/search.html**](https://docs.cs50.net/problems/sentiments/sentiments.html#code-templates-search-html-code)

Abra templates/search.html. Observe cómo esta plantilla representa el nombre de pantalla de un usuario, así como ese gráfico circular.

[**templates/layout.html**](https://docs.cs50.net/problems/sentiments/sentiments.html#code-templates-layout-html-code)

Abra templates/layout.html. Dentro de ahí hay una plantillla de la que index.html y search.html dependen. Ella apalanca a [Bootstrap](http://getbootstrap.com/) para anular la estética predeterminada de los navegadores

[**requirements.txt**](https://docs.cs50.net/problems/sentiments/sentiments.html#code-requirements-txt-code)

Abra requirements.txt (sin cambiarlo, aunque puede hacerlo más adelante si lo desea). Este archivo especifica las bibliotecas, una por línea, de las que depende toda esta funcionalidad.

[**Getting Started**](https://docs.cs50.net/problems/sentiments/sentiments.html#getting-started)

1. En una ventana terminal ejecute
2. cd ~/workspace/pset6/sentiments/

pip3 install --user -r requirements.txt

para instalar las dependencias de este programa.

1. Regístrese en Twitter en [twitter.com/signup](https://twitter.com/signup) si todavía usted no tiene una cuenta.
2. Visite [apps.twitter.com](https://apps.twitter.com/), haciéndo log in si se le pide, y de click **Create New App**.
   * Cualquiera (disponible) **Nombre** basta.
   * Cualquier (suficientemente larga) **Descripción** basta.
   * Para **Website**, ingrese **https://cs50.harvard.edu/** (o cualquier otro URL).
   * Deje **Callback URL** en blanco.
3. Haga click en **Create your Twitter application**. Ustede debería de ver "Your application has been created."
4. Haga click en **Keys and Access Tokens**.
5. Haga click en **modify app permissions**.
6. Seleccione **Read only**, luego haga click en **Update Settings**.
7. Haga click en **Keys and Access Tokens** de nuevo.
8. Resalte y copie el valor a la derecha de **Consumer Key (API Key)**.
9. En una ventana de terminal ejecute

export API\_KEY=value

en dónde value es ese (pegado) valor, sin ningún espacio inmediatamente antes o después de =.

1. Resalte y copie el valor a la derecho de **Consumer Secret (API Secret)**.
2. En una ventana de terminal ejecute

export API\_SECRET=value

En dónde value es ese (pegado) valor, sin ningún espacio inmediatamente antes o después de =.

Si usted cierra esa ventana de terminal y/o abre otra, usted necesitará repetir esos cinco pasos.

A continuación, intente correr

./smile

Para ver cómo funciona Tenga en cuenta que todas las palabras se clasificarán (por ahora!) Como neutrales debido a que se empotra 0 en analyze.py.

A continuación , trate de correr

flask run

y luego seleccione **CS50 IDE > Web Server** en CS50 IDE en la esquina izquierda superior. Busque el nombre de pantalla de un usuario, ¡y debería ver un gráfico! Por supuesto, todo es amarillo por ahora debido a 100.0 in application.py. Salag de Flask con control-c.

[**Especificación**](https://docs.cs50.net/problems/sentiments/sentiments.html#specification)

[**analyzer.py**](https://docs.cs50.net/problems/sentiments/sentiments.html#code-analyzer-py-code-2)

Complete la implementación de analyzer.py de tal manera que

* \_\_init\_\_ cargue palabras positivas y negativas en la memoria de forma que analyze pueda accesarlas y
* analyze analice los sentimientos de text, retornándo una calificación positiva si text es más positiva que negativa, una calificación negativa si text es más negativa que positiva, y 0 de otra manera, por lo cual esa calificación es computada a como sigue:
  + asigna a cada palabra en text un valor: 1 si la palabra es en positives, -1 si la palabra es en negatives, y 0 de otra manera.
  + considere la suma de esos valores para que sea la calificación completa del texto.

Por ejemplo, si text fuese "I love you" (y Analyzer fuesen instanciados con los valores por defecto para sus parámetros nombrados), entonces su puntaje sería

0 + 1 + 0 = 1, ya que

* "I" No es ninguno positive-words.txt o negative-words.txt,
* "love" es positive-words.txt, y
* "you" no está ni en positive-words.txt o negative-words.txt.

Basta decir que, [more sophisticated algorithms](https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html) existen, pero vamos a mantener las cosas simples!

[**tweets**](https://docs.cs50.net/problems/sentiments/sentiments.html#code-tweets-code-2)

Complete la implementation de main en tweets de manera que que el programa

* Acepte uno y solo un argumento de línea de comandos, el nombre de pantalla de un usuario de Twitter,
* Consulte la API de Twitter para los 50 tweets más recientes de un usuario,
* Analiza el sentimiento de cada uno de esos tweets, y
* Da como salida la puntuación de cada tweet y texto, coloreados en verde si es positive, rojo si es negativo, y amarillo de otra manera.

[**application.py**](https://docs.cs50.net/problems/sentiments/sentiments.html#code-application-py-code-2)

Complete la implementación de search en application.py de tal manera que la función

* Consulte la API de Twitter por los 100 últimos tweets de un usuario,
* Clasifique cada tweet como positivo, negativo, o neutral,
* genere un gráfico que describa con precision esos sentimientos en porcentajes.

Si un usuario ha tweeteado menos de 100 veces, clasifique tantos tweets como existan.

[**Walkthroughs**](https://docs.cs50.net/problems/sentiments/sentiments.html#walkthroughs)

[**Us**](https://docs.cs50.net/problems/sentiments/sentiments.html#usage)**o**

Su programa deberá comportarse tal como los ejemplos a continuación . Suponiendo que el texto subrayado es lo que algún usuario a tipeado.

$ ./smile

Usage: ./smile word

$ ./smile foo bar

Usage: ./smile word

$ ./smile love

:)

$ ./smile hate

:(

$ ./smile Stanford

:|

$ ./tweets

Usage: ./tweets @screen\_name

$ ./tweets @foo @bar

Usage: ./tweets @screen\_name

$ ./tweets @cs50

0 hello, @world

1 I love you, @world

-1 I hate you, @world

...

[**Testing**](https://docs.cs50.net/problems/sentiments/sentiments.html#testing)

No check50 para este! Pero aquí hay algunos nombres de pantalla reales en Twitter que pueden tener algunos sentimientos positivos o negativos!

* [cs50](https://twitter.com/cs50)
* [davidjmalan](https://twitter.com/davidjmalan)
* [DrJillStein](https://twitter.com/DrJillStein)
* [GovGaryJohnson](https://twitter.com/GovGaryJohnson)
* [HillaryClinton](https://twitter.com/HillaryClinton)
* [realDonaldTrump](https://twitter.com/realDonaldTrump)

[**Staff’s Solution**](https://docs.cs50.net/problems/sentiments/sentiments.html#staff-s-solution)

[**smile**](https://docs.cs50.net/problems/sentiments/sentiments.html#code-smile-code-2)

~cs50/pset6/smile

[**tweets**](https://docs.cs50.net/problems/sentiments/sentiments.html#code-tweets-code-3)

~cs50/pset6/tweets

[**Consejos**](https://docs.cs50.net/problems/sentiments/sentiments.html#hints)

[**analyzer.py**](https://docs.cs50.net/problems/sentiments/sentiments.html#code-analyzer-py-code-3)

Es probable que encuentre [nltk.tokenize.casual.TweetTokenizer](http://www.nltk.org/api/nltk.tokenize.html#module-nltk.tokenize.casual) muy interesante, que se puede usar para tokenizar un tweet (es decir, dividirlo en una lista de palabras) con un código como:

* tokenizer = nltk.tokenize.TweetTokenizer()

tokens = tokenizer.tokenize(tweet)

Por ejemplo, si el tweet es I love you, entonces tokens serán ["I", "love", "you"]. El tokenizer trata algunos signos de puntuación como tokens separados, así que no te preocupes si divide palabras como a+ (la cuál está en positive-words.txt) en dos tokens.

* Asegúrese de ignorar cualquier comentario o línea en blanco dentro de positives y negatives.
* Si desea que una variable sea accessible desde ambos \_\_init\_\_ y analyze, asegúrese de definirlo como una "instancia variable" dentro de self. Por ejemplo , si usted fuese a definir
* self.n = 42

dentro de \_init\_\_, entonces self.n también será accesible dentro de analyze.

* Es probable que encuentres [str.lower](https://docs.python.org/3/library/stdtypes.html#str.lower) muy interesante.
* Note que get\_user\_timeline retorna None en caso de error, como sucedería si un screen name no existe o los tweets de un screen name son privados.
* And here’s the [time-complexity (aka "Big O" or "Big Oh") of various operations in current CPython](https://wiki.python.org/moin/TimeComplexity), la implementación de Python que estamos usando (que es un intérprete llamado python, o realmente python3, que en sí mismo está escrito en C).

[**tweets**](https://docs.cs50.net/problems/sentiments/sentiments.html#code-tweets-code-4)

* Busca en smile algo de inspiración!

Porque tweets no termine en .py, CS50 IDE no sabe que es Código de Python, por lo que el resaltado de sintaxis no estará habilitado por defecto. Con el archivo abierto en una pestaña, cambie **Text** to **Python** en la esquina inferior derecha de la pestaña para habilitar

.

[**application.py**](https://docs.cs50.net/problems/sentiments/sentiments.html#code-application-py-code-3)

* Mira (de nuevo) tweets para un poco de inspiración!

[**FAQs**](https://docs.cs50.net/problems/sentiments/sentiments.html#faqs)

[**Could not build url for endpoint '/'**](https://docs.cs50.net/problems/sentiments/sentiments.html#could-not-build-url-for-endpoint)

Si encuentra que cuando intenta buscar en su aplicación Flask sin escribir nada en el campo de texto, obtienes un Could not build url for endpoint '/'., cambie la linea en application.py que lee redirect(url\_for("/"))que return redirect(url\_for("index")).

[**ImportError: No module named 'sqlalchemy'**](https://docs.cs50.net/problems/sentiments/sentiments.html#importerror-no-module-named-sqlalchemy)

Si aparece este error, ejecute

pip install --user sqlalchemy

para resolverlo!

[**twython.exceptions.TwythonAuthError: Twitter API returned a 401 (Unauthorized), An error occurred processing your request**](https://docs.cs50.net/problems/sentiments/sentiments.html#twython-exceptions-twythonautherror-twitter-api-returned-a-401-unauthorized-an-error-occurred-processing-your-request)

Si ve este error, lo más probable es que intente obtener tweets para un nombre de pantalla protegido (es decir, privado). No se preocupe, sin embargo. Puede suponer que solo probaremos su código con nombres de pantalla que no están protegidos.

[**twython.exceptions.TwythonError: Twitter API returned a 404 (Not Found), Sorry, that page does not exist**](https://docs.cs50.net/problems/sentiments/sentiments.html#twython-exceptions-twythonerror-twitter-api-returned-a-404-not-found-sorry-that-page-does-not-exist)

Si ve este error, ¡lo más probable es que intente obtener tweets para un nombre de pantalla que no existe! No se preocupe, sin embargo. Puede suponer que solo probaremos su código con los nombres de pantalla que existen.

[**TypeError: 'NoneType' object is not iterable**](https://docs.cs50.net/problems/sentiments/sentiments.html#typeerror-nonetype-object-is-not-iterable)

Si ve este error en un bucle for, asegúrese de estar iterando sobre una lista y no, por ejemplo, None. En particular, asegúrese de verificar el valor de retorno de get\_user\_timeline, que, según su implementación, puede devolver None en caso de error.

[**CHANGELOG**](https://docs.cs50.net/problems/sentiments/sentiments.html#changelog)

* 2016-10-27
  + Aclarado que search debe clasificar ⇐ 100 tweets.
* 2016-10-26
  + Agregado un consejo y una FAQ acerca de cómo get\_user\_timeline puede retornar None.
  + Aclarado que analyze toma una (potencialmente multi-palabra) str como un argumento, no solo como una palabra.
* 2016-10-21
  + Initial release.

[**Acknowledgements**](https://docs.cs50.net/problems/sentiments/sentiments.html#acknowledgements)

Agradecimiento especial a Aditi Muralidharan and John DeNero of UC Berkeley y a Minqing Hu and Bing Liu of the University of Illinois at Chicago!