PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2233 - Programación Avanzada (I/2016)
Tarea 7

1. Objetivos

- Interacción con una API para la creación de un programa.
- Sintetizar conocimientos del curso.

2. Problema

En esta tarea deberá crear una un programa a partir de la API de GitHub que permita realizar una serie de funcionalidades mediante una interfaz gráfica. En esta interfaz, un usuario deberá ser capaz de conectarse a su usuario de GitHub (real) y trabajar sobre este repositorio (creado para esta tarea).Una vez conectado se deben poder observar las opciones para las funcionalidades pedidas, realizarlas y para el caso de las estadísticas, desplegar los resultados obtenidos de la API en la interfaz. Para las otras dos funcionalidades (explicadas más adelante), el resultado deberá verse directamente en el repositorio.

3. Especificaciones

3.1. API

Como se mencionó anteriormente, para realizar esta tarea tendrá que usar la API de GitHub. Una API es una *Application Programming Interface* y consiste en un conjunto de recursos que permiten consumir funcionalidades de alguna plataforma en particular, de forma amigable con desarrolladores. Existen APIs de distintos servicios: Facebook, Amazon Web Services, Telegram, etc.. Como se dijo al principio, en esta tarea deberán usar la API de Github. Como se pueden imaginar, es de vital importancia que los clientes que consumen alguna API (es decir, ustedes) sepan exactamente qué parametros deben entregar al realizar una solicitud, así como también qué se espera que retornen los recursos de la API. Por ello es que deben estudiar y comprender la documentación de la API de Github. En particular, recomendamos fuertemente los siguientes enlaces para que se familiaricen con la estructura de la documentación: `ISSUES`, `REACTIONS`.[1]

Además, se encuentra a su disposición en el Syllabus una guía hecha por Antonio Ossa que explica a grandes rasgos la interacción de la API en Python.

[1] Estos enlaces son sólo puntos de partida en el viaje que deberán hacer a través de la documentación de la API de Github. Se espera que sean capaces de buscar información eficientemente en cualquier motor de búsqueda decente (Google, DuckDuckGo, etc.).

3.2. Funcionalidades

3.2.1. Creación y sucripción a *labels*

El programa será capaz de crear relaciones entre ciertos *labels* y usuarios de Github. Esta relación definirá a los usuarios que deban ser asignados a las *issues* con un determinado *label*. Al momento de crear las relaciones, el programa debe verificar que los usuarios que se relacionarán con dicha *label* puedan acceder al repositorio. Debido a que no existe forma de que Github avise al programa cuando se crean *issues* nuevas ni cuando se asignan *labels* a *issues* existentes, el programa deberá contar con algún mecanismo accionado por el usuario (puede ser un botón) para asignar los usuarios de la relación a las *issues* que correspondan. Para este item, considere solo las *issues* creadas después del 25 de Mayo.

3.2.2. Estadísticas

Dar la opción de desplegar distintas estadísticas:

- Dado un nombre de usuario, entregar el número de issues que abre.
- Dado un nombre de usuario, entregar el total de cada una de las reacciones que generan todas las issues que crea este mismo. Las reacciones existentes son:

content	emoji
+1	👍
-1	👎
laugh	😄
confused	😕
heart	❤️
hooray	🎉

- Dado un usuario, en cuantas issues participa (se dice que un usuario participa en una issue cuando la crea o comenta en ella).
- Dado un label, la cantidad de issues con ese label.

Para todas estas, deberá obtener todos los usuarios que tengan permisos en el repositorio (todos los alumnos de IIC2233) y todos los labels existentes en este dependiendo de la estadística a consultar.

3.2.3. *Issues* repetidas

Debe darse la opción de seleccionar una issue dentro de las que existan en el repositorio y un botón para comentar sus issues repetidas. A grandes rasgos deberán, entre todas las otras issues, detectar cuáles son las dos más parecidas y comentar su número en la issue original.

Para esto debe realizar las siguiente etapas:

1. **Transformar los datos de entrenamiento**

   a) Transformar cada issue (Título + Explicación) en una lista de palabras. Esta lista de palabras no puede contener puntuación ni stopwords[2] y cada palabra debe ser representada solo por su raíz. El módulo `utils` tiene dos métodos que lo ayudarán a hacer esto.

[2] Stopwords son palabras con una frecuencia mucho mayor al resto y que no aportan información significativa al sentido del texto. En general, son artículos y pronombres. Por ejemplo: el, ella, la, los, les, etc.

b) Transformar cada documento en un vector TF. Este vector indica cuántas veces se repite una palabra del diccionario de palabras en el documento. Por ejemplo: Sean `['hola', 'como', 'estas', 'bien']` las palabras del diccionario. El documento `['hola', 'bien']` se representaría de la forma `[1, 0, 0, 1]`.

c) Una vez que haya transformado todos los documentos a vectores TF debe calcular el vector IDF de cada uno de ellos.

2. **Buscar las issues más parecidas**

a) Transforma la issue nueva a la representación IDF.

b) Busca las dos issues de entrenamiento que tengan la menor distancia a la issue nueva. La menor distancia se calcula como la norma cuadrada ($|x - y|^2$) entre los dos vectores IDF.

Una vez que obtenga las dos issues debe generar un comentario dentro de la issue original con "#númeroissueparecida1 #issueparecida2"(formato que se les puede hacer conocido).

4. Restricciones y alcances

- Tu programa debe ser desarrollado en Python 3.4
- Esta tarea es estrictamente individual y está regida por el Código de Honor de la Escuela: Clickear para Leer.
- Su código debe seguir la guía de estilos `PEP8`
- Si no se encuentra especificado en el enunciado asuma que el uso de cualquier librería Python está prohibido. Pregunte por foro si se pueden usar librerías específicas.
- El ayudante puede castigar el puntaje[3] de tu tarea si le parece adecuado. Se recomienda ordenar el código y ser lo más claro y eficiente posible en la creación de algoritmos.
- Debe adjuntar un archivo `README.md` donde se comenten los alcances y el funcionamiento de su sistema (*i.e.* manual de usuario) de forma *concisa* y *clara*.
- Cree un módulo para cada conjunto de clases. Divídalas por las relaciones y los tipos que poseen en común.
- Cualquier aspecto no especificado queda a su criterio, siempre que no pase por sobre otro.

5. Entrega

- **Fecha/hora:** 24 de Junio - 23:59 hrs
- **Lugar:** GIT - Carpeta: Tareas/T07

Tareas que no cumplan con las restricciones señaladas en este enunciado tendrán la calificación mínima (1.0).

[3] Hasta −5 décimas.