Ontología de Twitter

Miguel Valencia Zurera
2º ciclo de Ingeniería en Informática

Felipe Geva Urbano

10/06/2014

FICHA DEL TRABAJO FINAL

Título del trabajo:	Ontología de Twitter
Nombre del autor:	Miguel Valencia Zurera
Nombre del consultor:	Felipe Geva Urbano
Fecha de entrega (mm/aaaa):	06/2014
Área del Trabajo Final:	XML y Web Semántica
Titulación:	2º ciclo de Eng. Informática

Resumen del Trabajo (máximo 250 palabras):

Twitter es un servicio que permite a sus usuarios enviar y publicar mensajes breves, generalmente solo de texto. Hoy en día, representa una fuente de información sobre las opiniones de millones de usuarios en relación a temas de actualidad.

Dentro del marco de la denominada Web semántica, el presente trabajo define una ontología para el modelo de información que se puede obtener del servicio Web Twitter.

Esta ontología esta implementada en el lenguaje OWL (Ontology Web Language), que es el más extendido. El lenguaje OWL ha sido diseñado para que las aplicaciones puedan procesar e integrar automaticamente el contenido el contenido de la información en la Web, en lugar de simplemente presentarlas para “consumo humano”. La ontología constituye un modelo con los conceptos mas importantes de Twitter y las relaciones entre ellos.

Adicionalmente, incluye un desarrollo software específico para poblar la ontología, es decir crear instancias a partir de la información que se puede extraer mediante la interfaz de comunicación de Twitter. Una vez poblada, los datos pueden ser explotados para obtener información valiosa.

Abstract (in English, 250 words or less):

Twitter is a service that allows its users to send and publish short messages, usually text only. Today, is a source of information on the opinions of millions of users in relation to current issues.

Within the framework of the so-called Semantic Web, this paper defines an ontology to model the information that can be obtained from the Twitter web service.

This ontology is implemented in the language OWL (Web Ontology Language) , which is the most widespread . The OWL language has been designed to enable applications to automatically process and integrate the content of the information content on the Web , rather than simply presenting them for " human consumption". The ontology is a model with the most important concepts of Twitter and the relationships between them .

Additionally, it includes a specific software development to populate the ontology, ie instantiated from the information that can be extracted via the communication interface of Twitter. Once populated, the data can be exploited to gain valuable information.

Palabras clave (entre 4 y 8):

Twitter, ontología, lenguaje OWL, conceptos, relaciones

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

Hoy en día, Twitter representa una fuente de información sobre las opiniones de sus usuarios en relación a temas de actualidad. El numero de usuarios[1] activos al mes es de 218 millones y envía 500 millones de tuits al día, es por tanto un recurso con mucho interés en ser analizado.

Actualmente cualquier contenido de Internet, incluido Twitter, se presenta de forma que podemos leerlo fácilmente. Además, existen los buscadores que nos permiten el acceso a casi cualquier contenido Web, pero aún así, existen ciertos problemas:

- Un alto número de positivos con baja precisión. Obtenemos demasiados resultados que hace que el resultado sea inútil.
- Los resultados son altamente sensibles al vocabulario utilizado.
- La información que deseamos se encuentra en varios documentos, lo que nos obliga a realizar múltiples búsquedas para encontrar todo lo que necesitamos.

En este contexto, aparece la necesidad de crear una ontología para almacenar la información que puede proporcionar la interfaz del servicio Twitter, de forma que podamos explotarla y obtener datos de interés.

En general, una ontología describe formalmente un dominio de conocimiento, en otras palabras, consiste en una lista finita de términos y de relaciones entre estos. Los términos están asociados a conceptos importantes del área sobre el que se establece la ontología. En definitiva la ontología nos permite:

- Superar las diferencias terminológicas
- Mejoran la precisión de las búsquedas Web
- Mejoran la navegación por los sitios Web

Etc.

1.2 Objetivos del Trabajo

Para un resultado satisfactorio del proyecto se plantean los siguientes tres objetivos, donde los dos primeros son de obligatorio cumplimiento y siendo el tercero opcional.

Crear una ontología.

Para desarrollar la ontología seguiremos los siguientes pasos:

- Definir las clases de la ontología.
- Organizar las clases jerárquicamente.
- Definir las propiedades de las clases y los valores permitidos para las mismas.
- Creación de instancias asignando valores a las propiedades.

En general, una ontología consiste en una lista finita de términos y de las relaciones entre estos, es por ello necesario un primer proceso de familiarización con estos términos que denoten conceptos importantes en este servicio. Por tanto, existe un proceso previo que será la curva de aprendizaje de la información que gestiona Twitter, y que podría englobar las siguientes actividades:

[1] TreceBits (2013). Twitter tiene 218 millones de usuarios activos al mes... [en línea]. http://www.trecebits.com/2013/10/04/twitter-tiene-218-millones-de-usuarios-activos-al-mes-y-envia-500-millones-de-tuits-al-dia/ [fecha de consulta: 05 de Marzo de 2014].

- Análisis del dominio de la ontología (en nuestro caso Twitter).
- Investigación de ontologías ya existentes relacionadas con el dominio.
- Estudio del marco de trabajo para la creación de la ontología (software Protégé).

Crear un programa para poblar la ontología.

Este programa debe usar la interfaz de que dispone el servicio Web Twitter para extraer información con la que poblar la ontología que se ha creado en la fase anterior. Para ello, distinguimos las siguientes actividades:

- Análisis de la Api de Twitter.
- Implementación del programa en Java mediante Eclipse.
- Carga de la ontología mediante el programa.
- Pruebas del proceso de carga.

Se incluye el último punto, para realizar un proceso de análisis sobre el resultado de la carga de forma que podamos verificar que tenemos un resultado correcto. Para ello se investigarán que métricas o pruebas son necesarias para certificar el resultado como correcto.

Mejoras.

El objetivo de esta fase es obtener información de los datos de la ontología aplicando reglas en base a alguna notación predeterminada. Para ello crear una serie de reglas, aplicarlas sobre la ontología y mostrar el resultado de una forma amigable al usuario, siguiendo los siguientes pasos:

- Análisis para seleccionar que tipo de reglas vamos a aceptar.
- Análisis del tipo de notación que debemos utilizar.
- Aplicación de reglas y visualización de los resultados.

En esta fase el objetivo es identificar que reglas se necesitan en la ontología y para ello debemos analizar que preguntas queremos hacer y que información queremos obtener.

Una vez conocemos las preguntas, debemos analizar el formato que vamos a utilizar para realizarlas. Se disponen de distintas opciones, por ejemplo; SWRL o bien DPL.

1.3 Enfoque y método seguido

En un principio, dada la evolución que ha tenido la Web semántica se espera encontrar una lista de ontologías bastante extensa, y por tanto parecía plausible comenzar a partir de una ya preestablecida y no de cero.

1.4 Planificación del proyecto

Las actividades están clasificadas en 3 fases:

- Crear la ontología. Donde tenemos como objetivo disponer de la ontología en un fichero en formato OWL.
- Crear el programa. Al finalizar esta fase debemos disponer de un programa con el cual poblar la ontología
- Incluir reglas. Donde estableceremos una serie de reglas para la ontología, y las aplicaremos para ver los resultados.

Cada fase se compone de una serie de tareas o actividades que permiten el desarrollo global del PFC. A continuación se incluye el desglose de estas tareas con una breve descripción de las mismas.

Tabla de actividades:

Actividad	Descripción	Duración
Análisis de los datos en Twitter	Obtener un vocabulario de términos asociados a conceptos de datos en Twitter y las relaciones entre ellos.	4 días
Reutilización de ontologías	Investigación de ontologías ya existentes relacionadas con el dominio, por si podemos partir de una previa. Esta tarea es recomendable porque hoy en día existe una amplia variedad de ontologías ya verificadas que nos pueden ayudar a comenzar con una base sólida de conocimiento.	4 días
Estudio del software Protégé	Análisis del marco de trabajo que se va a utilizar para crear la ontología, en nuestro caso el programa Protégé. Es importante que sepamos manejar la herramienta de trabajo para poder obtener todo su potencial.	2 días
Definir clases	Del vocabulario de términos asociados a Twitter se definen las clases mediante la ordenación de dichos términos en una taxonomía usando como marco el programa Protégé.	5 días
Establecer jerarquía	Se establece el orden de las clases para determinar que clase es subclase de quién.	3 días
Establecer propiedades	Se atribuyen las propiedades a las clases previamente definidas, teniendo en cuenta que existe herencia entre las clases y por tanto es conveniente que las propiedades se definan en la clase con más alta jerarquía que proceda.	4 días
Definir facetas	En este punto se enriquecen las propiedades añadiendo la siguiente información: cardinalidad, valores necesarios, y características relacionales.	3 días
Pruebas	Para comprobar la ontología deberemos poblar la misma creando algunas instancias y revisando que no existen anomalías.	3 días
Api de Twitter	Lectura de la documentación de la Api de Twitter con objeto de disponer del conocimiento suficiente para emprender con seguridad el desarrollo del programa.	5 días
Modelado del programa	Debido a que se trata de un programa pequeño, se agrupa en esta actividad todas las tareas previas a la implementación de cualquier producto software	4 días
Desarrollo	Desarrollo del programa en Java mediante el marco de trabajo Eclipse.	12 días
Poblar ontología	Fase en la que se pobla la ontología mediante la carga de información que se ha podido extraer con el programa. También podremos detectar inconsistencias tanto en la ontología como en el conjunto de instancias que se han usado para poblarla.	2 días
Análisis de reglas	Investigar que reglas queremos aplicar sobre la ontología.	3 días
Análisis de la notación	Investigar que notación se va a usar para la aplicación de las reglas definidas previamente.	5 días
Creación de las reglas	Crear reglas que actúen sobre los datos de la ontología.	12 días
Documentación	Documentar cada fase del PFC con objeto de realizar la memoria del proyecto.	-

Nota: Documentar, se trata de una actividad especial como se explica en el punto: Hitos importantes, ya que se desarrolla en paralelo con las demás actividades con la finalidad de ir documentando diariamente los avances del PFC. Una vez finalizado el PFC, a esta actividad se le incluye días adicionales con objeto de dar los últimos retoques a la memoria del proyecto y generar el video de presentación.

Diagrama de Gantt.

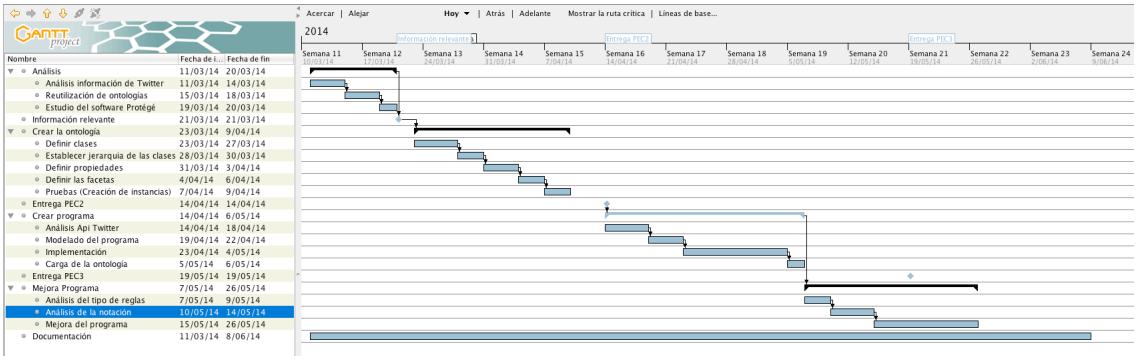


Ilustración 1

Horario de trabajo.

Aunque el cronograma se encuentra establecido en días, estos no corresponden a una jornada laboral de tipo normal, como es de suponer. El rango de horas asignado para este PFC, sería el siguiente:

Lunes y martes: 19:00 a 21:00
Miércoles, jueves y viernes: 18:00 a 21:00
Sábado y Domingo: 10:00 a 13:30 y 17:30 a 21:00

En consecuencia, tenemos 20 horas semanales para trabajar en este proyecto.

Para disponer de una estimación acorde al plan de trabajo supondremos que habrá un progreso estimado de 3 horas de trabajo diario para completar las 20 horas semanales.

De esta forma, si una actividad se ha calculado como 3 días de trabajo, aproximadamente podemos hablar de 9 horas de trabajo real.

Hitos importantes.

El resumen del calendario de trabajo se puede reducir en los siguientes puntos más importantes:

- Fase de análisis previa con el rango de tiempo: 11/03/2014 – 20/03/2014. Esta fase debe finalizar con la creación de un documento informal, con el vocabulario de términos (conceptos) de Twitter y la idea clara de si vamos a reutilizar o no una ontología existente.

- Hito: Información relevante. Validación del vocabulario de términos por parte del tutor. El objetivo es partir de una base de términos verificada.

- Fase de crear la ontología, con el rango de tiempo: 21/03/2014 – 10/04/2014. El resultado de esta fase debe ser la entrega de la PEC2 que finaliza el 14/04/2014.

- Fase de desarrollo del programa, con el rango de tiempo: 11/04/2014 – 03/05/2014.

- Fase de reglas, con el rango de tiempo: 04/05/2014 – 24/05/2014

- Documentación, con el rango de tiempo: 21/03/2014 – 30/05/2014, pero con el particular de que se trata de una actividad en paralelo con las demás y cuyo porcentaje de tiempo asignado sería de un 10% con respecto a la actividad del día. Es decir, un 10% de la actividad del día realizada se debe dedicar a documentar lo realizado.

Estos plazos son estimados, y pueden variar según las complicaciones que puedan surgir durante el desarrollo del PFC.

1.5 Breve sumario de productos obtenidos

El resultado de este proyecto debe permitir disponer de los siguientes productos:

En la carpeta con el nombre: ontología, se debe encontrar un fichero de extensión “owl” que contiene la definición de la ontología sobre el dominio de Twitter. El otro fichero con extensión “pprj” contiene la definición del proyecto en Protégé de la ontología.

En la carpeta plugin, se encuentra el programa compilado y empaquetado en formato zip, preparado para ser incluido en la aplicación de Protégé. Las instrucciones de uso se encuentran en el punto 6.1 de este documento.

En la carpeta fuentes, se encuentra el código fuente del programa preparado para ser incluido como un proyecto Maven en el entorno de trabajo Eclipse. También, se incluyen las librerías necesarias para su compilación.

Por último, se adjunta un documento Word, con la memoria del proyecto y un video de presentación del mismo.

1.6 Breve descripción de los otros capítulos de la memoria

Capitulo 2. Describe todo el proceso de creación de la ontología.
Capitulo 3. Es el análisis de requerimientos para el desarrollo del programa.
Capitulo 4. Describe el modelo de datos del programa.
Capitulo 5. Entra en detalle en el proceso de creación del programa.
Capitulo 6. Describe como se debe instalar y configurar el programa como un plugin para el entorno Protégé, y también, se incluye una descripción de uso del mismo.
Capitulo 7. Se entra en materia sobre la creación y uso de reglas SWRL como método de explotación de datos, y obtención de información de interés.

2. Creación de la ontología

"*Una ontología es una especificación explícita y formal de una conceptualización*"[2]. Este proyecto consiste, en una primera fase, en la creación de una ontología a partir del dominio de información; Twitter, donde se representará el conocimiento especializado pertinente a este servicio Web.

Para la representación semantica de los datos que se encuentran en el servicio Twitter, se hará uso del lenguaje OWL[3] (**O**ntology **W**eb **L**anguage), que es el más extendido. El lenguaje OWL ha sido diseñado para que las aplicaciones puedan procesar e integrar automaticamente el contenido el contenido de la información en la Web, en lugar de simplemente presentarlas para "consumo humano".

Para facilitar la tarea de creación de la ontología, se utilizará un editor de lenguaje OWL llamado Protégé[4] de código abierto. Este programa es uno de los editores más utilizados.

Twitter es un servicio que permite a sus usuarios enviar y publicar mensajes breves, generalmente solo de texto. En relación a este proyecto, una característica muy importante es que dispone de una interfaz de programación de aplicaciones (IPA) ó en ingles API (*Application Programming Interface)* abierta para todo tipo de desarrolladores, lo cual supone una gran ventaja para todos aquellos que quieran integrar Twitter como un servicio tanto en otras aplicaciones Web como en aplicaciones de escritorio o móviles.

Además Twitter dispone en su portal Web, de una sección con toda la documentación disponible en referencia a diversos temas, pero principalmente nos interesa el relacionado con los "objetos de la plataforma[5]". A partir de esta página Web comenzaremos la creación de nuestra ontología.

2.1 Elaborar una lista de términos

En este paso se debe elaborar una lista no estructurada de términos que se pretenden estén presentes en la ontología. Normalmente, los sustantivos constituyen la base de los nombres de clase (conceptos) y los verbos (o locuciones verbales) la base de las propiedades.

En este proceso se aconseja el uso de herramientas clásicas de la ingeniería del conocimiento, como el escalamiento (Laddering) o el análisis reticular (grid analisys).

Laddering[6].
El método de escalamiento se utiliza por lo general en combinación con otros métodos de obtención de conocimiento, tales como la clasificación de tarjetas. Los sujetos y objetos dentro de la ontología están interconectados con varios tipos de relaciones solicitada a los expertos del dominio a través de escalamiento, y el origen estructural de los sujetos y los objetos se construyen a través de la tarjeta de clasificación. El escalamiento se puede utilizar para tres propósitos principales:

---

[2] Definición de Gruber (1993: 199).

[3] Web Ontology Language. OWL es un lenguaje de Web semántica diseñado para representar el coocimiento rico y compleo de las cosas, los grupos de cosas y las relaciones entre estas. [en línea]. http://www.w3.or/2001/sw/wiki/OWL [fecha de consulta: 06 de Marzo de 2014].

[4] Protégé. Editor de ontologías y marco de trabajo para construir sistemas inteligentes. [en línea]. http://protege.stanford.edu/ [fecha de consulta: 06 de Marzo de 2014].

[5] Documentación de Twitter. A field guide to Twitter Platform objects. [en línea]. https://dev.twitter.com/docs/platform-objects [fecha de consulta: 07 de Abril de 2014].

[6] Laddering. The Internet encyclopaedia of personal construc psychology. [en línea]. http://www.pcp-net.org/encyclopaedia/laddering.html [fecha de consulta: 07 de Abril de 2014].

- Escalamiento para obtener subclases, tomando como ejemplo conceptos de Twitter; el sistema puede solicitar consejo a los usuarios para obtener alguna subclase de "Entities", a lo cual se puede responder: hashtag, y url. Este proceso podría continuar de forma recursiva con las subclases obtenidas, por ejemplo con "hashtag": se podría responder: reciente, y antiguo. Este cuestionario interactivo entre sistema y usuarios ayuda a construir las "escaleras".

- Escalamiento para obtener explicaciones. En este caso el sistema ya conoce los conceptos implicados en el dominio y propone elecciones entre los conceptos a los usuarios para poder registrar sus explicaciones, el objetivo es registrar la motivación de la elección de un concepto u otro.

- Escalamiento para obtener metas y valores. En esta etapa el sistema tiene la estructura y las explicaciones de los conceptos fundamentales, por lo que es posible descubrir las metas y los valores de cada usuario.

En este proyecto se hará uso de escalamiento para obtener las clases y subclases combinado con la información que se puede obtener de la documentación del servicio y refinandolo con ejemplos de datos concretos que se puedan obtener de la API de Twitter.

2.1.1 Primer paso. Conceptos iniciales.

La documentación de Twitter clasifica la información de su servicio en cuatro conceptos principales:

1. Tweets.
2. Users.
3. Entities.
4. Places.

Tweet[7]**.** Son el componente básico. Los usuarios crean tweets conocido como "actualizaciones de estado". Los tweets se pueden incrustar, responder a, ser favoritos, no favoritos, y ser borrados. Exactamente un tweet es un mensaje de texto plano de corta longitud, con un máximo de 140 caracteres, y dispone de las siguientes características o campos:

- contributors. Es una colección de usuarios (objeto User) que contribuyeron a la autoria del tweet.
- coordinates. Localización geográfica del tweet. El formato de estas coordenadas es un objeto de tipo geoJSON[8].
- create_at. Fecha de creación del tweet en formato UTC Time.
- entities. Lista de objetos "Entities" asociados al tweet (hashtags, media, urls, etc.).
- favorite_count. El numero de veces que este tweet ha sido marcado como favorito por los usuarios de twitter.
- favorited. Si el tweet ha sido favorito para el usuario autenticado.
- filter_level. A través de la etiqueta **'filter_level'** (filtro de nivel), los usuarios de Twitter podrán averiguar el nivel de calidad de sus tweets que se clasificarán como "ninguno", "bajo" o "medio". Las categorías Medio y Alto serán las que equivalen ahora a los tweets destacados.
- lang. Identificador del lenguaje (BCP 47) del tweet.
- place. Contiene un objeto Place (localización), que esta asociado al tweet.
- retweet_count. Numero de veces que este tweet ha sido retuiteado.
- retweeted. Indica si este tweet ha sido retuiteado por el usuario autenticado

---

[7] Tweets. A field guide to Twitter Platform objects. [en línea]. https://dev.twitter.com/docs/platform-objects/tweets [fecha de consulta: 07 de Abril de 2014].

[8] geoJSON. Es un formato para la codificación de una variedad de estructuras de datos geográficos. [en línea]. http://geojson.org/ [fecha de consulta: 07 de Abril de 2014].

- truncated. Indica si el texto del tweet ha sido truncado por exceder de 140 caracteres.
- user. El objeto usuario que envío este tweet.
- retweeted_status. Cuando un tweet es un retweet, este campo contiene el tweet original.
- id_str. Representación en formato texto del identificador único del tweet.
- source. Contiene información sobre la herramienta que fue usada para enviar el tweet. En caso de hacerse vía Web, contiene la URL de la misma.
- text. El texto del tweet en formato UTF-8.

En la página Web de la documentación de Twitter, que informa sobre este objeto, se declaran muchos más campos pero no se han tenido en cuenta los campos con la característica "Deprecated", porque es un estado que indica que la propiedad en cuestión debería ser evitada porque ha caído en desuso.

Tampoco se han tenido en cuenta los que tienen la característica "Unused", porque se supone que actualmente no están siendo usados y por tanto no podremos obtener nada de ellos.

Adicionalmente se han detectado campos en la documentación que dan información que ya puede ser recogida por otros, y por tanto redundarían en el modelo de datos, es el caso por ejemplo de:

- in_reply_to_screen_name. Si el tweet es un retweet, este campo contiene el screen_name del usuario al que pertenece el tweet original. Pero ya disponemos de "retweeted_status" que contiene toda la información del tweet original, por tanto este campo no aporta nada nuevo.
- id. Representación numérica del identificador único del tweet.
- In_reply_to_status_id. Representación numérica del tweet original, en el caso de ser un retweet.
- Etc...

User[9]. Los usuarios pueden ser cualquier persona o cualquier cosa. Ellos tuitean, siguen, crean listas, tienen una línea de tiempo (home_timeline), pueden ser mencionados, y se les puede consultar de forma masiva. Disponen de las siguientes características:

- contributors_enabled. Indica que el usuario tiene una cuenta con "modo contribución" activada permitiendo a los tweets creados por el usuario ser co-escritos por otra cuenta.
- created_at. Fecha de creación de la cuenta del usuario en Twitter.
- description. Texto con la descripción de la cuenta del usuario.
- entities. En el caso del objeto usuario se usan para describir las URL que aparecen en los campos definidos por el usuario URL de perfil y descripción.
- favourites_count. El numero de tweets que el usuario ha marcado como favoritos.
- followers_count. El numero de seguidores que tiene actualmente la cuenta del usuario.
- friends_count. El número de usuarios de esta cuenta está siguiendo (AKA sus "seguidores"). Se trata del numero de usuarios que este usuario sigue, es decir el numero de usuarios de los cuales este usuario es seguidor.
- geo_enabled. Indica si el usuario tiene activado el geotagging en sus tweets. Geotagging, una tecnología que nos permitirá etiquetar nuestras imágenes con una referencia al lugar donde han sido tomadas.
- is_translator. Indica si el usuario es participante de la comunidad de traducción de Twitter.
- lang. Código BCP 47 del idioma declarado por el usuario.
- listed_count. Numero de listas publicas de las que el usuario es miembro

[9] Users. A field guide to Twitter Platform objects. [en línea]. https://dev.twitter.com/docs/platform-objects/users [fecha de consulta: 07 de Abril de 2014].

- location Texto con la localización de usuario, puede ser interpretada por el servicio de búsqueda con lo que podría ser un dato interesante para conocer datos estadísticos por localización.
- name. Nombre del usuario limitado a 20 caracteres.
- protected. Cuando esta a activado (valor: true) quiere decir que el usuario ha elegido proteger sus tweets.
- screen_name. Representa el alias del usuario (máximo 15 caracteres)
- verified, Indica que la cuenta ha sido verificada.
- id_str. Representación en formato texto del identificador único del usuario.
- statuses_count. El numero tweets (incluidos retweets) que ha enviado el usuario.
- url. Contiene una URL asociada al perfil del usuario.

Se han aplicado las mismas reglas, que en el caso de los tweets, para obviar campos del objeto: user. Adicionalmente, también se han desestimado los campos que estaban asociados a informaciones sobre el perfil del usuario, como imágenes de fondo, temas, colores, etc., que no se han considerado interesantes para el modelo conceptual.

Entities[10]. Las entidades proporcionan metadatos e información contextual adicional sobre el contenido publicado en Twitter. Suelen estar asociadas a algún otro de los conceptos de Twitter, usualmente: Tweet, y User.

Las "entidades" están clasificadas en varios subtipos, aunque la documentación indica: *en general, es mejor considerar un campo nulo, un conjunto vacío, y la ausencia de un campo como la misma cosa*. En consecuencia, por ejemplo si un tweet no trae la referencia de la entidad subtipo: "hashtag", no quiere decir que no pueda venir en otro tweet, así pues consideraremos en el modelo todos los posibles casos de una entidad, que son:

- hashtag. Es una palabra que va precedida del símbolo # (almohadilla). Los hashtags permiten diferenciar, destacar y agrupar una palabra o tópico especifico en esta red social. Con esto se consigue crear una etiqueta para aquellas palabras y así poder agruparlas y separarlas de otros temas que incluyen el mismo término, pero que estén usándolo con un sentido diferente al que se desea otorgarle.
- media. Representa cualquier elemento multimedia (imágenes, videos, etc..).
- url. Representa cualquier objeto de tipo URL[11].
- user_mention. Representa una lista de usuarios que aparecen en el campo text (en el caso de tweets, y retweets) o bien en la descripción (en el caso de: user).
- symbol. Representa una lista de símbolos financieros que deben comenzar con el carácter del dolar ($) y que son extraídos del texto del tweet (esta entidad solo aplica a los tweets).

Place[12]. Este objeto determina una localización con sus correspondientes coordenadas espaciales. Este tipo de objetos pueden ser asociados a los tuits mediante un identificador (denominado place_id). Los lugares no tienen porque indicar la ubicación del tuit, sino simplemente estar asociados de alguna forma con el.

---

[10] Entities. A field guide to Twitter Platform objects. [en línea]. https://dev.twitter.com/docs/platform-objects/entities [fecha de consulta: 07 de Abril de 2014].

[11] Uniform Resource Locator (URL). RFC 1738: syntax and semantics of formalized information for location and access of resources via the Internet. [en línea]. http://www.ietf.org/rfc/rfc1738.txt [fecha de consulta: 07 de Abril de 2014].

[12] Places. A field guide to Twitter Platform objects. [en línea]. https://dev.twitter.com/docs/platform-objects/places [fecha de consulta: 07 de Abril de 2014].

- attributes. Se trata de un objeto que contiene información asociada al lugar. Se trata de una lista compuesta por el par: clave-valor.
- bounding_box. Una lista con las coordenadas de los limites del lugar
- country. Nombre del país
- country_code. Código del país, representado mediante dos letras.
- full_name. Nombre completo del lugar.
- id. Representa el identificador único del objeto: place.
- name: nombre corto del lugar, útil para una lectura comprensible.
- place_type. Tipo de localización que representa el lugar.
- url. Texto que representa una URL asociada al lugar.

2.1.2 Segundo paso. Subclases.

En este punto, ya hemos obtenido una primera lista de términos de los conceptos clave que maneja Twitter. De esta lista, los que se pueden asociar a tipo de datos básicos (String, int, float, datetime, etc.) ya no interesan, ahora se entra en detalle de aquellos que son objetos mas complejos.

En relación a los Tweets.

- contributors. Es una colección de usuarios que contribuye al tweet, la cual según la documentación solamente dispone de tres atributos: id, id_str, y screen_name. Aunque solamente dispone de 3 campos del global que representa a un usuario, se trata en definitiva de objetos: User, así pues "contributors" es una lista de objetos User.

- coordinates. Representa una lista de un nuevo tipo, que denominaremos: geoJSON.

- retweet. Cuando un tweet es una respuesta o replica a otro tweet se le denomina retweet, y es un caso especial de tweet porque esta asociado a otro tweet generado previamente, pero no se puede considerar un subtipo porque realmente un retweet no dispone de propiedades que lo diferencien de otro tweet, así pues se trata de una relación o propiedad que pueden tener los tweets.

- entities. Representa una colección de objetos: Entities, es decir es una asociación entre los tweets y los entities.

- user. Representa al usuario que creó el tweet. Se trata de una asociación directa con un único objeto: User.

- place. Representa una localización relacionada con el tweet. Se trata de una asociación directa con un único objeto: Place.

geoJSON. Una estructura de datos GeoJSON completa es siempre un objeto (en términos JSON), el cual esta compuesto por una colección de pares de nombre / valor - también llamada miembros. Para cada miembro, el nombre es siempre una cadena. Valores de los miembros son una cadena, número, objeto, matriz o uno de los literales: true, false, y nulo. Dispone de los siguientes atributos:

- latitude. Representa la distancia angular entre la línea ecuatorial (el ecuador), y un punto determinado de la Tierra.
- longitude. Representa la distancia angular entre un punto dado de la superficie terrestre y el meridiano que se tome como 0°.
- type. Es un texto que representa el tipo de geometría asociada a los puntos anteriores.

Así pues en referencia al objeto Tweet, podemos afirmar las siguientes conclusiones:

- Tiene o puede tener contribuyentes. Una lista de usuarios que contribuyen a la autoria del tweet.
- Tiene o puede tener coordenadas. Una lista de objetos geoJSON que determinan una ubicación.
- Tiene o puede tener entidades. Una lista de metadatos asociados al tweet.
- Esta o no asociado a un lugar. Un objeto de tipo: Place
- Puede o no ser un retweet. Es decir esta asociado a un único tweet en una relación hijo:padre, donde un hijo solamente puede tener un padre, pero un padre puede tener multiples hijos.

En relación a los Users.

- entities. Representa una colección de objetos: Entities, es decir es una asociación entre los usuarios y los entities. En este caso la documentación de twitter especifica que las únicas entidades asociadas a un usuario son de tipo: URL, que pueden encontrarse en su perfil.

- status. Este campo contiene el tweet o retweet mas reciente del usuario, pero se especifica claramente que esta información no puede ser obtenida siempre y por tanto dicho campo puede ser omitido, nulo o estar vacío. En cualquier caso es una información que puede ser redundante, ya que el sistema "stream Api" de Twitter permite obtener los tweets que se van generando (así poblaremos la ontología, como ya se verá mas adelante), de los cuales extraemos sus usuarios creadores, por tanto ya dispondremos del último tweet asociado al usuario.

Así pues en referencia al objeto User, podemos afirmar las siguientes conclusiones:

- Tiene o puede tener entidades. Una lista de metadatos asociados al usuario.

En relación a las Entidades (Entities).

Según la documentación de Twitter un objeto entidad solamente puede ser uno de los siguientes tipos: hashtag, url, media, user_mention y symbol. Cada uno de ellos, tiene sus propias características, que son las siguientes:

HashTag.

- text. Texto del hashtag sin el carácter almohadilla (#).

Media.

- display_url. Una URL de acceso al elemento para los usuarios.
- expanded_url. La versión expandida del contenido de display_url.
- id_str. Identificador único de tipo texto del elemento multimedia
- media_url. La URL que apunta directamente al elemento.
- sizes. Representa un objeto complejo que determina el tamaño del elemento.
- type. Tipo del elemento

Url.

- display_url. Versión de la URL para visualizar en los clientes.
- expanded_url. La versión expandida del contenido de display_url.

User_mention.

- id_str. Identificador único, en formato texto de un objeto: User.
- name. Nombre del usuario.
- screen_name. Alias del usuario.

Symbol.

- text. Texto del símbolo sin el carácter del dólar ($).

En relación a la ontología tendremos en cuenta las siguientes afirmaciones que aparecen en la documentación de Twitter, con respecto a las entidades y sus relaciones con los demás objetos:

- Los Tweets pueden incluir todos los tipos de entidades que se han descrito.
- Para los usuarios los tipos: hashtag y user_mention no son aplicables, si en cambio el tipo: url, que permite identificar URLs que se encuentran en el perfil del usuario.
- Si una entidad es de un subtipo determinado, por ejemplo: hashtag, este no puede ser de ningún otro subtipo al mismo tiempo. Esta regla aplica a todos los subtipos.
- El subtipo User_mention, en realidad identifica a un usuario, aunque la Api de Twitter solamente ofrece tres de los campos que globalmente representan a un usuario.

Por último, se detalla el objeto complejo: size del subtipo Media, que dispone de las siguientes características:

- height. Altura en pixeles
- width. Anchura en pixeles
- resize. Indica el método usado para el tamaño que se esta usando. Solamente dispone de dos valores posibles: fit y crop. El primero indica que se ha adaptado manteniendo su relación con el tamaño original, el segundo que se ha recortado para obtener el tamaño.
- type. Indica el tipo de versión: thumb, large, medium, o small.

<u>En relación a los lugares (Places)</u>.

- attributes. Este campo contiene información asociada al lugar. Se trata de una lista compuesta por el par: clave-valor. Las claves posibles son:
    - street_address. Dirección
    - locality. Nombre de la ciudad
    - region. Región administrativa
    - iso3. El código del país
    - postal_code. Código postal
    - phone. Teléfono.
    - twister. Alias del usuario
    - url. URL del lugar
    - app: Un identificador o lista separada por comas de identificadores que representan el lugar en la base de datos de aplicaciones.

- bounding_box. Contiene una lista con las coordenadas de los limites del lugar, representadas en con los datos de longitudes, latitudes y tipo de geometría, en consecuencia se pueden asociar al tipo geoJSON, que anteriormente se ha detallado.

2.2 Definir la taxonomía

Una vez identificados los términos, en esta etapa se deben ordenar en una taxonomía. Existen diferentes opiniones sobre si es mas fiable o eficaz que la organización se haga con el ordenamiento ascendente o descendente, pero en mi opinión en este dominio no creo que afecte, si es más importante en este paso determinar la jerarquía de los términos, es decir conocer que clase es subclase de quién.

- Tweet
- User
- Entity
    - HashTag
    - Media
    - Url
    - Symbol
- Place
- GeoJSON
- GeoPlaceAttribute
- SizeMediaEntity

Reglas que se han tenido en cuenta:

- Se ha utilizado la nomenclatura en inglés para utilizar términos que se pueden asociar directamente en la documentación de Twitter.
- Los nombres de las clases serán siempre en singular
- Los nombres de las clases deben empezar siempre con la primera letra en mayúscula.
- Cuando el nombre de la clase se compongan de dos palabras estas iran unidas y la primera letra de la segunda palabra irá siempre en mayúsculas.

2.3 Definir propiedades

En esta etapa se definen las propiedades que representarán las características, atributos y relaciones de las clases.

Se deben tener muy en cuenta a que clases asignar las propiedades ya que si A es una subclase de B, cada propiedad de B también será valida para A. Debido a esta característica de herencia, las propiedades se deben establecer en la clase mas alta de la jerarquía, siempre que las condiciones lo permitan.

Así mismo, al asignar las propiedades se aconseja siempre asignar el dominio y rango de las mismas. Un dominio[13] de propiedad reduce los individuos a los que puede aplicarse la propiedad. Si una propiedad relaciona un individuo con otro individuo, y la propiedad tiene una clase como uno de sus dominios, entonces el individuo debe pertenecer a esa clase.
El rango[14] de una propiedad reduce los individuos que una propiedad puede tener como su valor. Si una propiedad relaciona a un individuo con otro individuo, y ésta como rango a una clase, entonces el otro individuo debe pertenecer a dicha clase.

En este contexto definimos las propiedades de las clases citadas en el punto anterior, tendiendo en cuenta que pueden ser de dos tipos:

---

[13] rdfs:domain. Lenguaje de Ontologías Web (OWL) Vista General. [en línea]. http://www.w3.org/2007/089/OWL-Overview-es.html [fecha de consulta: 07 de Abril de 2014].

[14] rdfs:range. Lenguaje de Ontologías Web (OWL) Vista General. [en línea]. http://www.w3.org/2007/089/OWL-Overview-es.html [fecha de consulta: 07 de Abril de 2014].

1. Objects properties. Determina una relación entre instancias de dos clases.
2. DataType properties. Determinan relaciones entre instancias de clases, literales RDF, y tipos de datos básicos.

2.3.1 Propiedades de la clase Tweet

En base a los términos detectados en fases anteriores se pueden definir las siguientes propiedades que afectan a esta clase, y que agrupamos en la siguiente tabla:

Propiedad	Tipo	Rango	Descripción
createAt	DataType	dateTime	Fecha de creación del tweet
favorited	DataType	boolean	Si el tweet ha sido marcado como favorito
filterLevel	DataType	string	Nivel de calidad de los tweets
idStr	DataType	string	Identificador único
lang	DataType	string	Identificador del lenguaje del tweet (en formato BCP 47)
retweetCount	DataType	int	Numero de veces que el tweet ha sido retuiteado
retweeted	DataType	boolean	Si el tweet es un retweet
source	DataType	string	Identifica la herramienta usada para enviar el tweet
text	DataType	string	Texto del tweet
truncate	DataType	boolean	Si el texto del tuit ha sido truncado por exceder de 140 caracteres
hasContributors	Object	User	Lista de usuarios que han contribuido al tweet
hasCoordinates	Object	GeoJSON	Localización geográfica del tweet
hasEntities	Object	Entity	Lista de metadatos asociados al tweet
hasPlace	Object	Place	Ubicación o lugar asociado al tweet
hasReply	Object	Tweet	Lista de tweets que han replicado a este tweet
isCreated	Object	User	Usuario que ha creado el tweet
isReply	Object	Tweet	Contiene el tweet original al que este tweet ha replicado
mentionUser	Object	User	Lista de usuarios que se mencionan en el tweet

Entre los términos había dos campos: id e id_str que representan lo mismo, es decir un identificador único para el tweet, así pues se ha seleccionado solamente uno como propiedad para esta clase. Se ha elegido: id_str porque la documentación de Twitter indica que este identificador puede ser un numero bastante extenso, y por ello es mas manejable en formato texto.

Cabe decir, que todas las propiedades de la clase Tweet, tienen, como mínimo, el dominio: Tweet.

2.3.2 Propiedades de la clase User

Las propiedades de los usuarios se agrupan en la siguiente tabla:

Propiedad	Tipo	Rango	Descripción
contributorsEnabled	DataType	boolean	Indica si la cuenta esta en modo contribución
createdAt	DataType	dateTime	Fecha de alta del usuario en Twitter
description	DataType	string	Descripción de la cuenta de usuario
favouritesCount	DataType	int	Numero de tweets marcados como favoritos
followersCount	DataType	int	Numero de seguidores del usuario
friendsCount	DataType	int	Numero de usuarios al que sigue este usuario
geoEnabled	DataType	boolean	Indica si el geotagging esta activado
idStr	DataType	string	Identificador único del usuario
lang	DataType	string	Idioma declarado por el usuario
listedCount	DataType	int	Numero de listas públicas de las que es miembro el usuario
location	DataType	string	Localización del usuario

name	DataType	string	Nombre del usuario
protected	DataType	boolean	Indica si la protección de tweets esta activada
screenName	DataType	string	Alias del usuario
statusesCount	DataType	int	Numero de tweets y retweets enviados por el usuario
url	DataType	string	Url asociada al perfil del usuario
verified	DataType	boolean	Indica si la cuenta esta verificada

2.3.3 Propiedades de la clase Entity

En principio no existen propiedades que puedan ser comunes a todos los subtipos que engloba esta clase, asi que pasamos directamente a cada una de sus subclases:

Subclase	Propiedad	Tipo	Rango	Descripción
Hashtag	textHashtag	DataType	string	Texto del hashtag
Media	displayUrl	DataType	string	Url para utilizar en los clientes
	expandedUrl	DataType	string	Versión extendida del contenido de displayUrl
	idStr	DataType	string	Identificador único
	mediaUrl	DataType	string	Url que apunta directamente al contenido multimedia
	typeMedia	DataType	string	Tipo de elemento multimedia
	url	DataType	string	
Symbol	textSymbol	DataType	string	Texto del símbolo
Url	displayUrl	DataType	string	Url para utilizar en los clientes
	expandedUrl	DataType	string	Versión extendida del contenido de displayUrl
	url	DataType	string	

En este caso, el atributo de texto que se repite en varias de las subclases, se ha declarado uno propio para cada uno debido a que semanticamente tanto el contenido como uso y funcionalidad del mismo servia a propositos distintos en cada una de ellas, por ejemplo:

- El texto del hashtag identifica un nombre o frase asociado a un tema que se referencia en el tweet.
- El texto del symbol representa por el contrario un simbolo financiero.

Por otra lado, si ha reutilizado la propiedad “url”, ya que en todas las clases y subclases donde se utiliza el sentido de la propiedad siempre es el mismo.

2.3.4 Propiedades de la clase Place

Las propiedades de esta clase se agrupan en la siguiente tabla:

Propiedad	Tipo	Rango	Descripción
country	DataType	String	Nombre del país
countryCode	DataType	String	Código del país
countryfullName	DataType	String	Nombre largo del país
idStr	DataType	String	Identificador del lugar
typePlace	DataType	string	El tipo de localización del lugar
url	DataType	string	Una URL asociado al lugar
hasAttributes	Object	GeoPlaceAttribute	Lista de propiedades del lugar
hasBoundingBox	Object	GeoJSON	Lista de puntos geográficos que determinan el lugar

2.3.5 Propiedades de la clase GeoJSON

Las propiedades de esta clase se agrupan en la siguiente tabla:

Propiedad	Tipo	Rango	Descripción
geoLatitude	DataType	float	Latitud del punto geográfico
geoLongitude	DataType	float	Longitud del punto geográfico
geoType	DataType	String	Tipo de estructura geométrica

2.3.6 Propiedades de la clase GeoPlaceAttribute

Las propiedades de esta clase se agrupan en la siguiente tabla:

Propiedad	Tipo	Rango	Descripción
geoPlaceAttributeKey	DataType	String	Clave del atributo asociado a un lugar (Place)
geoPlaceAttributeValue	DataType	String	Valor del atributo

2.3.7 Propiedades de la clase SizeMediaEntity

Las propiedades de esta clase se agrupan en la siguiente tabla:

Propiedad	Tipo	Rango	Descripción
height	DataType	int	Altura en pixeles de la medida
width	DataType	Int	Ancho en pixeles de la medida
typeSize	DataType	String	Tipo de tamaño
resize	DataType	String	Método usado para obtener este tamaño

2.4 Definir las facetas

En esta etapa se debe establecer las siguientes tres características:

1. Cardinalidad. Donde se deben especificar la cantidad de valores que pueden tener las propiedades.

2. Valores necesarios. Algunas propiedades solamente pueden moverse en cierto rango de valores.

3. Características relacionales. Donde se definen las propiedades de: simetría, transitividad, propiedades inversas y valores funcionales.

En definitiva ahora se profundiza en más detalle las tablas presentadas en el punto anterior, pero además se va a integrar todo en una única tabla para detectar posibles anomalías, como; clases o propiedades que son sinónimas, bucles, etc.

Clase	Propiedad	Rango	Cardinalidad	Otras restricciones
Tweet	createAt	dateTime	single	
	favoriteCount	int	single	
	favorited	boolean	single	
	filterLevel	string	single	oneOf: none, low, medium, high
	idStr	string	single	
	lang	string	single	
	retweetCount	int	single	

	retweeted	boolean	single	
	source	string	single	
	text	string	single	
	truncate	boolean	single	
	hasContributors	User	multiple	
	hasCoordinates	GeoJSON	multiple	
	hasEntities	Entity	multiple	
	hasPlace	Place	single	
	hasReply	Tweet	multiple	inverseFunctional, inverse: isReply
	isCreated	User	single	
	isReply	Tweet	single	Transitive, inverse: hasReply
	mentionUser	User	multiple	
User	contributorsEnabled	boolean	single	
	createdAt	dateTime	single	
	description	string	single	
	favouritesCount	int	single	
	followersCount	int	single	
	friendsCount	int	single	
	geoEnabled	boolean	single	
	idStr	string	single	
	lang	string	single	
	listedCount	int	single	
	location	string	single	
	name	string	single	
	protected	boolean	single	
	screenName	string	single	
	statusesCount	int	single	
	url	string	single	
	verified	boolean	single	
Entity				
Hashtag	textHashtag	string	single	
Media	displayUrl	string	single	
	expandedUrl	string	single	
	idStr	string	single	
	mediaUrl	string	single	
	typeMedia	string	single	oneOf: photo
	url	string	single	
Symbol	textSymbol	string	single	
Url	displayUrl	string	single	
	expandedUrl	string	single	
	url	string	single	
Place	country	string	single	
	countryCode	string	single	
	countryfullName	string	single	
	idStr	string	single	
	typePlace	string	single	
	url	string	single	
	hasAttributes	GeoPlaceAttribute	multiple	
	hasBoundingBox	GeoJSON	multiple	
GeoJSON	geoLatitude	float	single	
	geoLongitude	float	single	
	geoType	string	single	oneOf: Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon
GeoPlaceAttribute	geoPlaceAttributeKey	string	single	oneOf: locality, region, iso3, postal_code, phone, twitter, url, app:id, street_address
	geoPlaceAttributeValue	string	single	

SizeMediaEntity	height	int	single	
	width	int	single	
	resize	string	single	oneOf: crop, fit
	typeSize	string	single	oneOf: medium, thumb, small, large

La propiedad más compleja, según representa la tabla, es: isReply cuyo significado semántico sería: El tweet es una respuesta a otro tweet (o como se denomina en Twitter: un retweet), y se le atribuyen las características de:

- Functional (cardinalidad: single). Si una propiedad es "Functional" (única), no tendrá más de un valor para cada individuo. En este caso, un tweet solamente puede tener un tweet origen.

- Transtitive. Cuando una propiedad es transitiva, si el par (x, y) es una instancia de la propiedad transitiva P, y el par (y, z) es otra instancia de la propiedad transitiva P, entonces el par (x, z) también es una instancia de P. En este caso, cuando un tweet A es un retweet de C, si D es un retweet de A, también lo será de C.

- Inverse. Es posible definir una propiedad como la inversa de otra propiedad. En este caso, hasReply es la inversa de isReply porque:

    - Un tweet A puede ser una respuesta (isReply) a otro tweet B
    - Un tweet B puede tener múltiples respuestas (hasReply) de otros tweets; A, C,..

Por otra parte, también se pensó como propiedad una característica de los usuarios que son los seguidores: un usuario A podía ser seguidor de otro usuario B (y también existía su inversa). Esta sería una propiedad muy interesante, pero la documentación de Twitter y ejemplos que proporcionaba su "stream Api", no indicaban el modo de obtener esta información para la ontología, y es por esta razón que no se han incluido.

Una vez que disponemos de clases con una jerarquia y las propiedades que la relacionan, se puede generar un modelo en forma de un grafico relacional de forma que visualmente se tenga una primera impresión de la ontología:

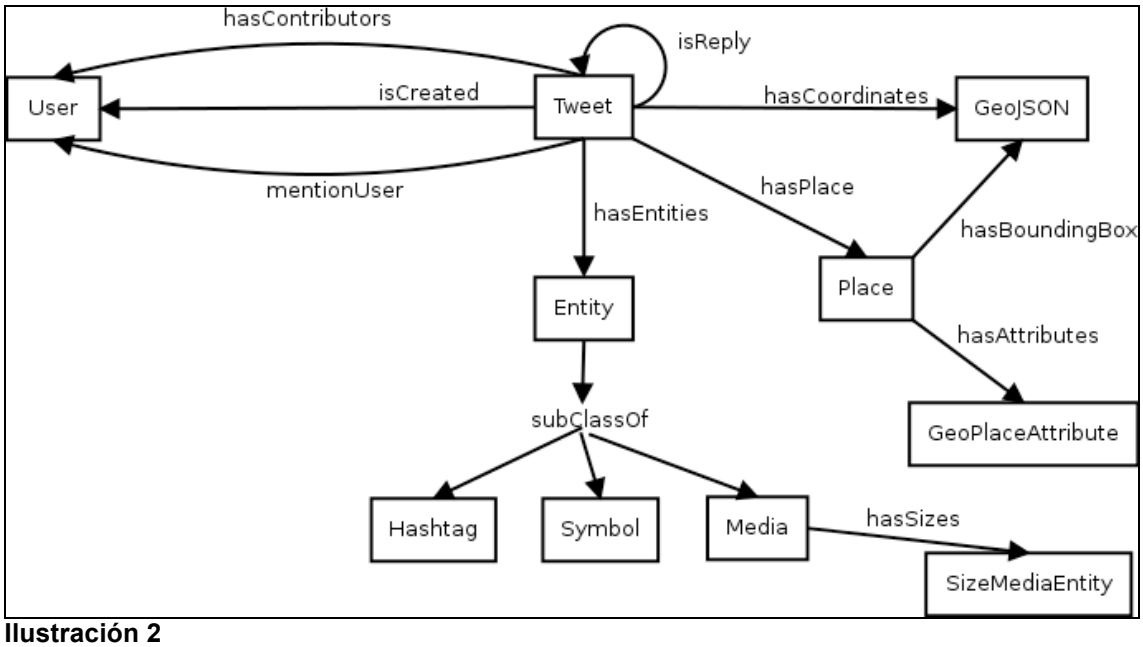


Ilustración 2

3. Análisis del desarrollo del programa

Una vez se dispone de una ontología en el marco de trabajo de la aplicación Protégé, se inicia esta fase del desarrollo de un programa que tiene como objetivo poblar la ontología que se ha creado anteriormente. Para ello se conectará a Twitter de forma que obtenga los datos necesarios para este proceso.

Para la implementación del programa se ha utilizado el entorno de trabajo “Eclipse”, y siguiendo las pautas de un proyecto Maven de forma que fuese más fácil crear la estructura del proyecto y manejar las dependencias hacías otras librerías.

El proyecto se basa en tres pilares fundamentales:

1. Integración con Protégé.
2. Conexión a Twitter.
3. Procesamiento de los mensajes.

3.2 Análisis de requerimientos

En este punto se deben describir todos los requerimientos que debe satisfacer el programa con objeto de determinar que se debe hacer y por tanto cumplir con las expectativas del mismo.

Un requerimiento “es una característica del sistema o una descripción de algo que el sistema es capaz de hacer con el objeto de satisfacer el propósito del sistema”. A la hora de establecer estos requisitos es importante que cumplan los siguientes criterios:

- Deben escribirse en términos que se puedan entender fácilmente.
- Deben ser consistentes, es decir, ni ambiguos ni tampoco conflictivos.
- Los requerimientos serán clasificados como: funcionales y no funcionales.
- Los requerimientos deben ser verificables mediante algún tipo de prueba.

Cabe señalar que el conjunto de requisitos deben servir a tres propósitos:

1. Permitir que el programador explique como ha entendido lo que se pretende del programa.
2. Indicar que funcionalidades y características va a tener el resultado.
3. Establecer una relación entre requisitos y pruebas que certifique o mida la validez del resultado.

3.2.1 Requisitos funcionales

Estos requerimientos deben describir la funcionalidad o servicios que se espera del programa, tanto por parte de los usuarios como del sistema donde se vaya a utilizar.

RF01. El programa debe poder conectar a Twitter para extraer información de los Tweets de los usuarios de este servicio Web.
Descripción: Se realizará un pequeño análisis de las diferentes posibilidades de conexión a Twitter, para confrontar ventajas e inconvenientes de cada uno y finalmente realizar la elección mas apropiada.

RF02. El programa debe poder filtrar la información de Twitter.
Descripción: Se debe habilitar una sección de filtros con los que poder seleccionar o más bien acotar el rango de información que debe ser utilizada para poblar la ontología.

RF03. El usuario debe poder arrancar y parar el proceso de poblar la ontología.
Descripción: El programa no debe ser automático a la hora de poblar la ontología sino que debe ser el usuario, previa configuración del mismo, quien tenga arrancar y parar el proceso.

3.2.2 Requisitos no funcionales

Son los requerimientos que no se refieren directamente a las funciones especificas que entrega el sistema, sino a las propiedades del programa, tales como fiabilidad, tiempo de respuesta, etc.

RN01. La aplicación debe integrarse con el framework Protege 3.5.
Descripción: Con objeto de minimizar la curva de aprendizaje de los usuarios con el aplicativo y hacerlo intuitivo con respecto al framework de trabajo que es la herramienta Protégé para el modelado de ontologías, parece lógico plantear el desarrollo del programa como un plugin del sistema Protégé.

RN02. Los parámetros de conexión a Twitter deben poder ser configurables.
Descripción: La finalidad es que cualquier usuario que active el plugin pueda establecer la conexión a Twitter de forma personal desde la herramienta.

3.3 Casos de uso

Aunque el programa a diseñar en bastante sencillo en cuanto a su uso, es importante siempre definir los posibles casos, es decir describir el comportamiento del programa ante las situaciones mas comunes al ser utilizado por los usuarios.

En primer lugar se incluirán los diagramas de casos de uso, y a continuación se describen las especificaciones suplementarias para capturar detalles de requisitos que caen fuera del ámbito de los diagramas.

Caso de uso 1.

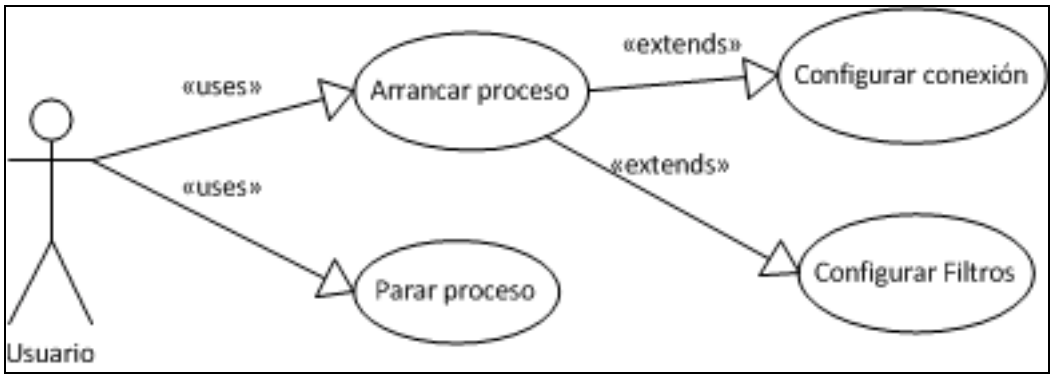


Ilustración 3

La aplicación permite dos operaciones al usuario: Arrancar proceso y Parar proceso. La primera operación lleva implícito dos tareas:

a) Configurar conexión. Es una tarea obligatoria y necesaria para establecer la conexión a Twitter.
b) Configurar filtros. Es una tarea opcional, donde el usuario puede establecer condiciones que los mensajes deben cumplir.

3.4 Diagrama de secuencia

Mediante este diagrama se muestra la interacción del conjunto de objetos de la aplicación a través del tiempo, existiendo uno por cada caso de uso. El objetivo del diagrama es disponer de una descripción del caso de uso como una secuencia de pasos, de forma que permita descubrir que objetos son necesarios para que se puedan seguir los pasos.

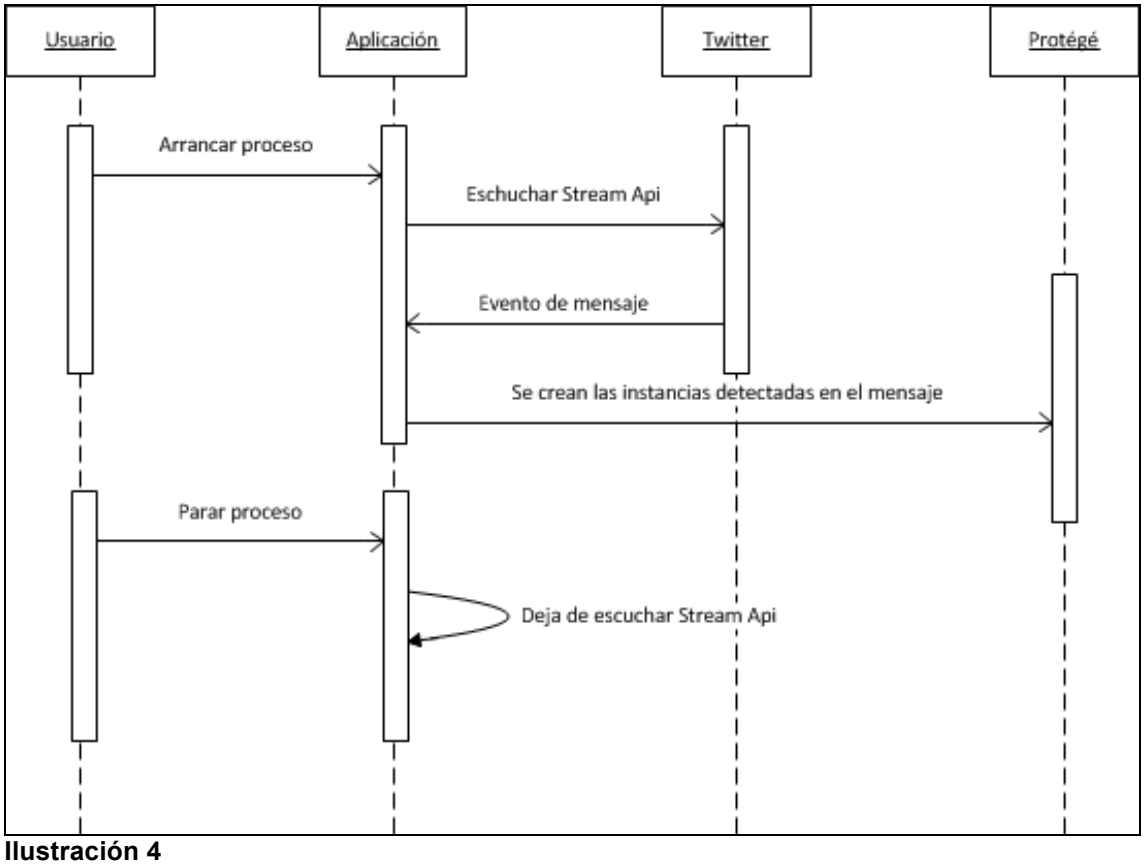


Ilustración 4

La secuencia es sencilla, y el proceso más complejo es cuando se crean las instancias porque no es algo fijo sino que depende de cada mensaje. Existirán mensajes con más y otros con menos información y en base a lo cual se crearan más o menos instancias.

3.5 Integración con Protégé

Para integrar la aplicación con Protégé, se debía desarrollar como un plugin para esta herramienta, con lo cual el primer paso tenia que ser conocer otros plugins y como funcionaban. Por suerte, Protégé dispone de una página de ayuda sobre su libreria[15] de plugins, que definen como:

La librería de Plugins para Protégé ofrece un lugar conveniente para la comunidad de encontrar código abierto y plugins comerciales que mejoran la aplicación.

[15] Wiki Protégé. Welcome to the Protege Plugin Library!. [en línea]. http://protegewiki.stanford.edu/wiki/Protege_Plugin_Library [fecha de consulta: 09 de Mayo de 2014].

Los plugins son clasificados en algunos de los siguientes tipos:

- Api. Permiten extender la funcionalidad básica de Protégé.
- Application. Son aplicaciones externas que usan como base a Protégé.
- Backend. Permiten ampliar la funcionalidad sobre los formatos de almacenamiento de las ontologías.
- Import. Permiten crear bases de conocimiento en Protégé con información de otros programas o fuentes externas.
- Export. Permiten exportar las bases de conocimiento de Protégé a otros formatos.
- Project. Permiten manipular la interfaz de usuario y los archivos de proyecto en distintos puntos de su ciclo de vida.
- Reasoner. Este tipo de plugin fue introducido en la versión 4 y queda descartado.
- Slot Widget. Permiten ampliar el grupo de componentes que gestionan la funcionalidad de las propiedades.
- Tab Widget. Permiten crear nuevas pestañas en la interfaz de usuario y aportando nueva funcionalidad a la herramienta.
- View. Este tipo de plugin fue introducido en la versión 4 y queda descartado.

A simple vista, el tipo más adecuado para este proyecto parecía el: Tab Widget, así que se escogió uno de estos plugins como marco de referencia. Se analizó el plugin: DataMaster[16], que permite importar datos de una base de datos relacional, de forma abstracta es similar al objetivo de nuestro proyecto.

La creación de un plugin en Protégé debe contemplar las siguientes características:

- Debe disponer de un archivo Manifest.mf que debe indicar la clase principal de la aplicación y el tipo de plugin.
- Debe disponer de un archivo denominado: plugin.properties que contendrá las dependencias[17] del plugin con otros plugins.
- Como el tipo de plugin seleccionado ha sido el Tab Widget, la clase principal debía extender de “edu.stanford.smi.protege.widget.AbstractTabWidget”.

Hay que hacer mención especial a la interfaz gráfica de usuario que se utiliza en Protégé, AWT (**A**bstract **W**indows **T**oolkit) es un kit de herramientas de gráficos, interfaz de usuario, y sistema de ventanas independiente de la plataforma original de Java. El uso de estas herramientas ha requerido una pequeña curva de aprendizaje para poder gestionar las ventanas, botones y eventos necesarios para la aplicación.

3.6 Conexión a Twitter

Como ya se ha comentado anteriormente, se ha utilizado la librería de streaming hosebird para conectar a Twitter y recoger los mensajes de los tweets.

La librería consta de dos módulos, el hbc-core and hbc-twitter4j el cuál es el encargado de empujar el mensaje desde la api e implementa una cola de mensaje para que las aplicaciones de consumo puedan sondear el mensaje a Twitter. El segundo módulo permite conectar a los usuarios el modelo de datos Twitter4J en la librería principal para analizar los tweets y mostrar finalmente el mensaje en las aplicaciones del usuario final.

---

[16] DataMaster Plugin. DataMaster is a Protege plug-in for importing schema structure and data from relational databases into Protege. [en línea]. http://protegewiki.stanford.edu/wiki/DataMaster [fecha de consulta: 09 de Mayo de 2014].

[17] Wiki Protégé. Instructions for declaring dependencies between Protege plug-ins. [en línea]. http://protegewiki.stanford.edu/wiki/PluginDependencies [fecha de consulta: 09 de Mayo de 2014].

Pero antes de usar esta librería era necesario disponer de unas credenciales para poder conectar a Twitter, es decir claves de acceso. Para "engancharnos" necesitamos crear una aplicación en Twitter mediante los siguientes pasos:

- Previamente debemos disponer de una cuenta en Twitter.
- Ahora, entramos a dev.twitter.com y haciendo click en SignIn, nos logamos.
- Vamos al mismo botón y hacemos click en Mis Aplicaciones. Le damos al "Create an Application", y llenamos toda la información (callback URL lo dejamos en blanco) y click a Crear.
- Por último, sólo queda ingresar a la nueva "app" y generar las claves.

En este punto ya debemos disponer del token de acceso OAuth que nos permitirá la autenticación para usar la Api Streaming de Twitter, y estará formado por los siguientes cuatro parámetros:

a) consumer key
b) consumer secret
c) token
d) secret

3.7 Procesamiento de los mensajes

Antes de comenzar a tratar los mensajes de Twitter, era necesario analizar dos condiciones:

a) Que tipo de mensajes nos podíamos encontrar
b) Formato de los mensajes

En la documentación de Twitter se haya un documento exhaustivo en relación al primer punto que establece los siguientes tipos de mensajes[18]:

- Public stream messages
- User stream messages
- Site stream messages

Esta página ofrece una perspectiva de los mensajes que nos podíamos ir encontrando al capturar los eventos que se van produciendo en Twitter, de forma que podamos evitar lo que no nos interesa y podamos tratar lo que si interesa.

En resumen, en este proyecto solamente se han tratado los mensajes que indicaban un nuevo tweet, y por tanto representan información susceptible de ser incluida en la ontología. Para identificar este tipo de mensajes se debe buscar el campo "id_str" que contiene el identificador único del tweet. Si el mensaje contiene el campo id_str en su posición correcta significa que tenemos un nuevo tweet y podemos procesarlo, en caso contrario no usamos la información del mensaje.

En relación al formato de los mensajes, Twitter trabaja con texto en formato JSON[19], con lo que fue necesario realizar una pequeña investigación para determinar la mejor forma de trabajar con este formato.

---

[18] Documentation Twitter. Streaming messages types. [en línea]. https://dev.twitter.com/docs/streaming-apis/messages [fecha de consulta: 09 de Mayo de 2014].

[19] Wikipedia. JSON. [en línea]. http://es.wikipedia.org/wiki/JSON [fecha de consulta: 09 de Mayo de 2014].

JSON, acrónimo de JavaScript Object Notation, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

Existen multitud de librerías para trabajar con este formato en diferentes entornos, en este proyecto el análisis se centro en trabajar usando java, así pues las más mencionadas fueron:

- quick-json Parser. https://code.google.com/p/quick-json/
- Json in java. http://www.json.org/java/index.html
- Java API for JSON Processing (JSON-P). https://json-processing-spec.java.net/

Aunque todas ofrecían buenas perspectivas, me decidí por la última por los siguientes motivos:

- Esta construida usando Maven, se encuentra en el repositorio central y por tanto la integración con el proyecto sería inmediata.
- Es la versión oficial de Java para el tratamiento de textos en formato JSON.
- Esta basada en el estándar JSR 353

4. Modelo de datos

Aunque en términos estrictos, el modelo de datos de la aplicación es la ontología creada para este programa, en la propia aplicación se han creado una serie de objetos que conforman un modelo de datos propiamente dicho y que permiten almacenar la información obtenida de Twitter y que posteriormente se pasa a la ontología.

El siguiente gráfico muestra la relación de clases del programa:

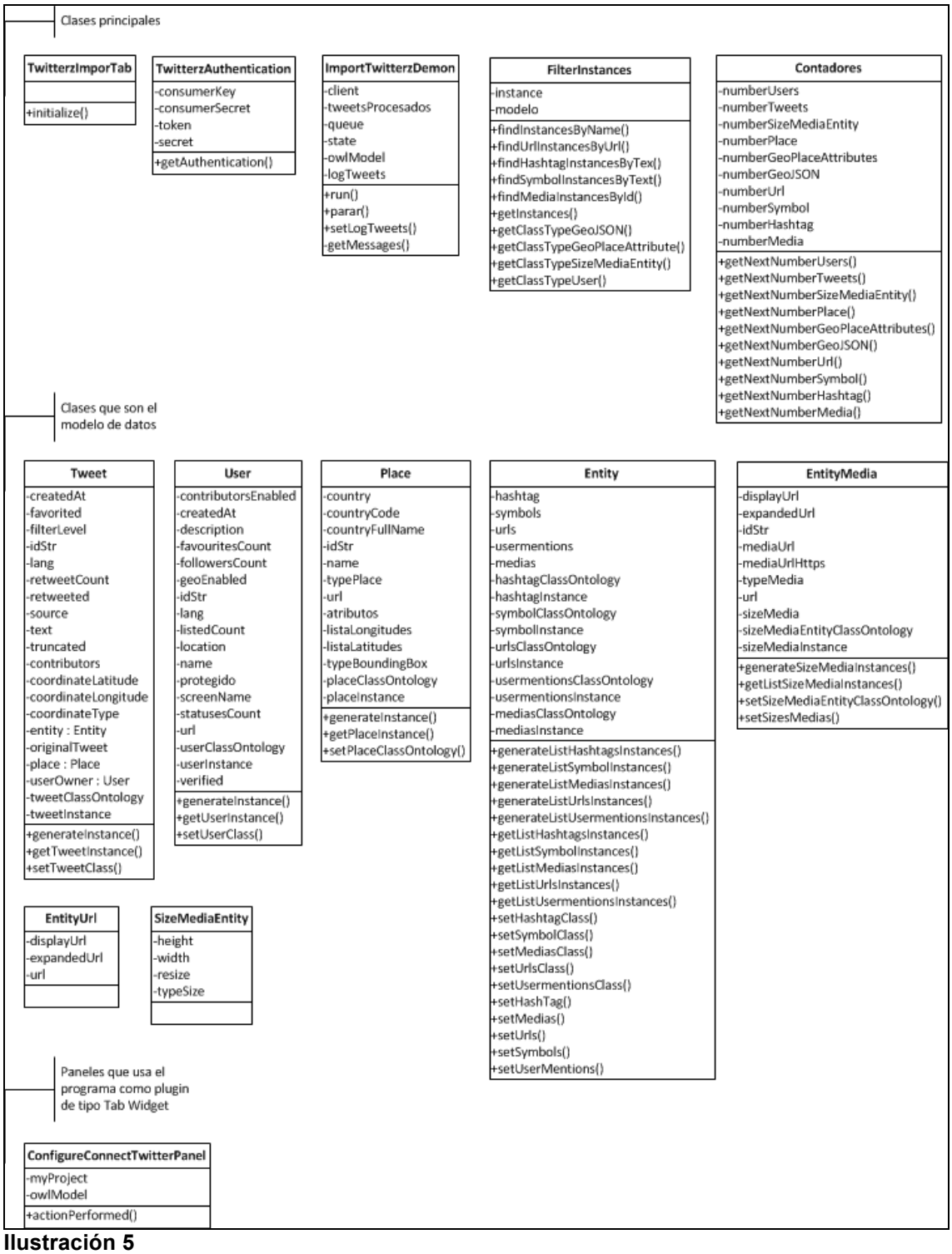


Ilustración 5

Nota: En el diagrama solamente se presentan los atributos y métodos más representativos de cada clase.

Como se indica en el gráfico, las clases se distribuyen en tres zonas:

- Modelo. Donde se agrupan todas las clases que permiten almacenar la información que se extrae de Twitter.
- Paneles. Las clases que generan la pestaña que aparece en Protégé.
- Principal. El resto de clases.

A continuación se entra en detalle sobre cada una:

TwitterzImportTab.

Esta clase extiende de: AbstractTabWidget y por tanto es la base para la creación de un plugin del tipo Tab Widget. Su funcionamiento es muy simple, ya que solo dispone de un método llamado: "initialize", donde se activa el panel o paneles que forman la pestaña en Protégé.

Contadores.

Esta clase esta creada siguiendo el patrón de diseño: singleton[20], para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella.

La clase contiene un atributo de tipo entero, y asociado a cada elemento del modelo que puede generar una instancia de forma que cada uno de ellos pueda disponer de un contador.

FilterInstances.

En esta clase se agrupan distintos métodos para realizar búsquedas sobre el conjunto de instancias creadas en Protégé. El objetivo de estas búsquedas es intentar no crear dos veces el mismo contenido, así pues cuando se usa un método para buscar existen siempre dos posibilidades;

a) que devuelva null, quiere decir que no ha encontrado la instancia
b) que devuelva un objeto de tipo: OWLIndividual, que es la instancia encontrada

Existen diferentes métodos porque se ha utilizado diferentes variaciones de código para crear las instancias, con la finalidad de probar diversas formas de hacer lo mismo. Por ejemplo:

- Para los usuarios (User) se ha utilizado como nombre el identificador único, que es el campo: idSrt, así pues para buscar entre las instancias existe una forma rápida de localizar usuarios que es: modelo.getOWLIndividual(nombre)
- Sin embargo para otros tipos de instancias no existía un atributo a usar tan evidente y por tanto se uso el nombre de la clase seguida del contador, así pues para buscar coincidencias se hacia bajo otro campo, por ejemplo para las instancias de la clase Url se obtenía todo el conjunto y se iteraba buscando y comparando bajo la propiedad: url.

TwitterzAuthentication.

Es una clase de ayuda para crear un objeto del tipo: com.twitter.hbc.httpclient.auth.Authenticacion

Este objeto permite la autenticación en Twitter.

---

[20] Wikipedia. Singleton. [en línea]. http://es.wikipedia.org/wiki/Singleton [fecha de consulta: 09 de Mayo de 2014].

ImportTwitterzDemon.

Esta clase es el principal motor del proceso de mensajes provenientes de Twitter. Extiende de Thread, con la finalidad de lanzar el proceso de forma independiente a la aplicación.

Dispone de dos constructores, en función del filtro que se vaya a usar para recoger los mensajes. En el constructor se crea el cliente de conexión a Twitter.

En el método run, es donde se arranca el hilo de procesamiento, en la siguiente secuencia:

- El cliente se conecta a Twitter.
- Se recogen los mensajes en una cola con un tiempo máximo de espera de 5 segundos.
- Si existe algún mensaje en la cola, se extrae y se procesa para ver si corresponde con un Tweet valido.
- En caso afirmativo, se generan las instancias. Este proceso se hace en cadena puesto que al generar la instancia del objeto Tweet, se generan el resto de instancias de otras clases si se han detectado los datos correspondientes.
- Este proceso se repite hasta que se ejecuta el método parar.

ConfigureConnectTwitterPanel.

Esta clase contiene toda la lógica que genera la interfaz de usuario en Protégé, es decir la pestaña que aparece cuando se activa este plugin. El constructor es donde se inicia el panel y mediante el método: actionPerformed se recogen y se tratan todos los eventos que se producen en el mismo.

Tweet.

Es la clase más importante del modelo de datos, y que agrupa toda la información que se puede recoger de un mensaje de Twitter asociado a un tweet. Cuando se detecta un mensaje en el proceso se le pasa como objeto de tipo JSON al constructor de esta clase, que inmediatamente lo analiza en busca de los campos que componen un tweet, esto significa que busca todas las propiedades declaradas en el modelo de la ontología.
Desde el constructor se van llamando al resto de clases del modelo si se encuentra en el mensaje la información asociada a cada una de ellas.

El otro método importante de esta clase es: generateInstance(), el cual genera una nueva instancia de la clase Tweet en la ontología. Este proceso, puede generar otras instancias asociadas al resto de clases de la ontología.

User.

Esta clase recoge los datos asociados a un usuario de Twitter. Al igual que la clase Tweet, contiene un constructor que rellena las propiedades a través de un análisis de un objeto JSON y dispone del método: generateInstance para crear una instancia de la clase User.

Place.

Esta clase esta asociada al objeto Place de Twitter, donde hace referencia a lugares determinados a través de coordenadas geográficas y que están asociados con el tweet de alguna forma. Al generarse una instancia de este tipo de objeto, existe la posibilidad de que se creen también, instancias del tipo: GeoJSON, y GeoPlaceAttribute.

Entity.

Esta clase recoge la lista de entidades que pueden estar asociadas a un tweet, y que pueden ser: Hashtag, Symbol, Medias, Urls, y Usermentions. Cuando se detecta que un tweet dispone de una o varias “Entities”, entonces se crean colecciones agrupándolas por su tipo, y

que permiten posteriormente generar las instancias correspondientes para ser incluidas como referencia en la instancia del tweet.

Las entidades de tipo: Hashtag, y Symbol no requieren disponer de clases propias ya que simplemente son un texto. La entidad Usermentions, son usuarios aunque con solo unos pocos datos de referencia, pero se puede usar la clase User. El resto de entidades disponen de sus propias clases que las definen.

EntityMedia.

Esta clase permite el tratamiento de entidades de tipo: Media. Este tipo representa elementos multimedia asociados al tweet.

EntityUrl.

Esta clase permite el tratamiento de entidades de tipo: Url. Este tipo representa Urls incluidas en el texto de un tweet, o bien asociadas a campos contextuales del usuario/s asociados al tweet.

SizeMediaEntity.

Esta clase permite definir un objeto asociado a las entidades de tipo: Media, y que representa un tamaño de los disponibles para el elemento multimedia.

5. Funcionamiento del programa

5.1 Proceso de análisis del mensaje

Cuando se detecta un mensaje (ver ejemplos en Anexo 11.4), el primer paso es crear el objeto JSON del mensaje y pasarlo al constructor de la clase Tweet, el cual sigue la siguiente secuencia:

- Obtener los atributos básicos del Tweet que corresponden a las "datatype properties" de la clase en la ontología, es decir: createAt, idStr, lang, etc...
- Se inicializan los objetos que pueden contener a los "object properties" de la clase Tweet.
- Se buscan los datos asociados al usuario que ha creado el tweet, si encuentra la entrada correspondiente genera un objeto: User pasando el JSON de estos datos.
- Se busca la entrada de las entidades asociadas al tweet, si esta entrada contiene datos entonces se crea un objeto Entity. Esta llamada puede a su vez generar objetos de tipo: EntityMedia, EntityUrls, y User.
- Se busca si el tweet tiene contribuidores, en cuyo caso se genera una lista de objetos User con estos usuarios.
- Se busca si la entrada correspondiente a "Coordenadas" contiene datos.
- Se analiza si el mensaje tiene asociado un lugar, y en caso afirmativo se crea un objeto de tipo: Place con esta información.
- Por ultimo, si existe la entrada "retweeted_status" y no esta vacía, quiere decir que el tweet es un retweet y dicha entrada contiene el tweet original, con lo que se genera un nuevo objeto Tweet con el tweet original.

Al finalizar el proceso de construcción del Tweet, dicho objeto ya dispone en distintas variables de toda la información que ha leído del objeto JSON con el que fue instanciado.

5.2 Proceso de generación de instancias

Antes de comenzar este proceso, cualquier tipo de instancia necesita, siempre, que se invoque un método que le permite conocer la estructura de la clase en la que se basa la instancia que se va a generar. Esta información se obtiene siempre del modelo que a su vez se obtiene del proyecto que es información asociada a cualquier plugin en Protégé.

Entonces, cualquier clase del modelo siempre dispone de un método llamado: setXxxClass, donde Xxx representa el nombre de la clase y que permite establecer una variable del tipo: *edu.stanford.smi.protegex.owl.model.OWLNamedClass*, que nos permite obtener toda la información de estructura de la clase, y de esta forma podemos acceder a las propiedades que tienen las instancias.

Una vez establecido el objeto: OWLNamedClass, se puede proceder a llamar al método: generateInstance() de la clase Tweet, pero el funcionamiento del mismo es muy similar en todas las clases del modelo y sigue la siguiente secuencia:

- Antes de crear una instancia se comprueba que no exista ya en la ontología. Para este proceso se hace uso del objeto FilterInstances, que contiene métodos para realizar estas búsquedas. Si la instancia existe nos devolverá dicho objeto y el proceso habrá terminado, en caso contrario seguimos.
- A partir del modelo se obtienen el Set de propiedades de la clase para la cual se va a generar la instancia, en principio: Tweet
- Se crea la instancia en blanco.
- Se recorren las propiedades de las instancias y se van asignando las propiedades con la información que está almacenada en el objeto.

- Cuando se han recorrido todas las propiedades la instancia ya esta completa y el proceso ha finalizado.

Para todas las clases el proceso de generación siempre es el mismo, y que podemos resumir en los pasos:

1. Chequear si existe la instancia.
2. Crear la instancia.
3. Asignar valores a sus propiedades.

Posteriormente existen pequeñas variaciones en el código en función de la clase, propiedades, tipos de valores, etc., pero principalmente podemos destacar:

Valores múltiples. Algunas propiedades pueden aceptar múltiples valores, en estos casos normalmente se utiliza el mismo método: *setPropertyValue* de la instancia pero pasándole un objeto de tipo Collection. Este procedimiento tiene su excepción en el caso de la propiedad: hasEntities de Tweet, en donde hubo de usarse: *addPropertyValue* por cada uno de los posibles valores porque asignar una colección que podía incluir distintos tipos de objetos producía un error.

Valores reales. Hay propiedades que fueron definidas en la ontología con el rango: float. En un principio se pensó que las variables definidas como Double en java servirían para guardar este tipo de datos, pero al almacenar el valor de la propiedad en la instancia daban error, asi pues este tipo de valores se debían pasar a float, mediante la expresión java: *variable.floatValue()*.

6. Configuración y gestión del programa

A continuación se incluye una breve descripción del plugin: Twitterz para Protégé, como se instala y como se usa.

6.1 Instalar el plugin

Junto con este documento se incluye un empaquetado en formato zip que incluye el este plugin para la aplicación Protégé versión 3.5. Dicho fichero se debe descomprimir en la carpeta plugins de la aplicación que debe generar una carpeta denominada: uoc.mvalencia.pfc y que debe contener los siguientes dos archivos:

- plugin.properties
- twitterz-0.0.1-SNAPSHOT.jar

Una vez están comprobados que tenemos esa estructura con los archivos indicados, podemos arrancar la aplicación del protégé. Para activar el plugin seleccionamos: Project | Configure, y debe aparecer la siguiente ventana:

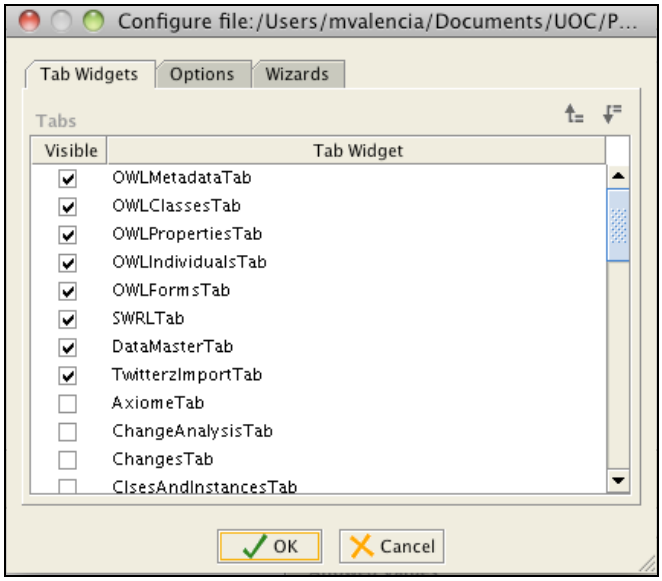


Ilustración 6

En la pestaña “Tab Widgets” debemos marcar: TwitterzImportTab, lo cual hará que se active la pestaña: Tweetz.

6.2 Funcionamiento del plugin

Una vez activado el plugin se habilita una pestaña, en la aplicación Protégé, denominada: Tweetz, donde podemos interactuar con el mismo y que presenta el siguiente formato:

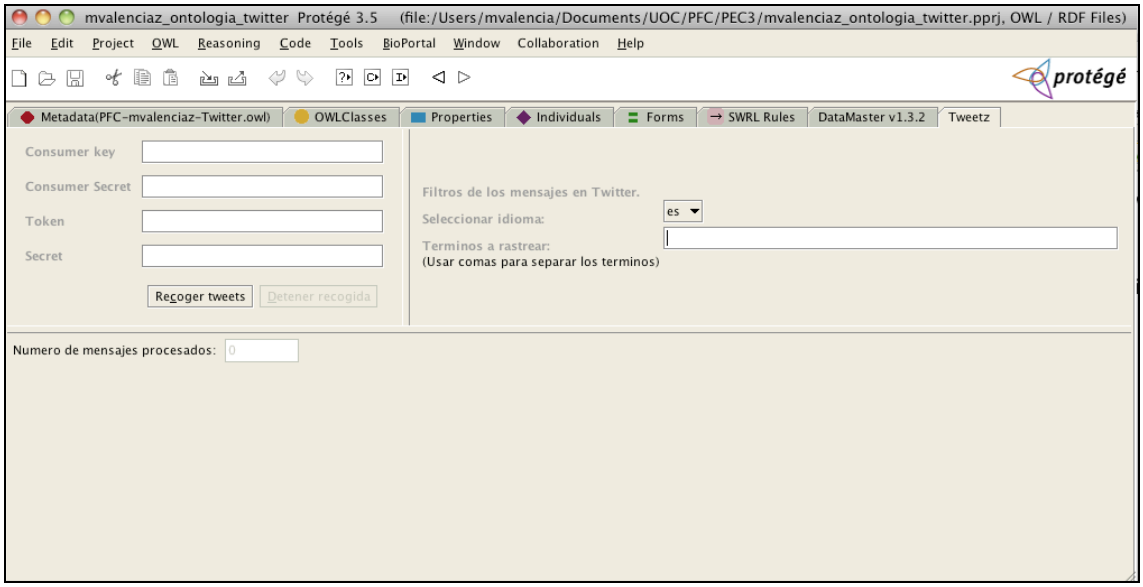


Ilustración 7

El panel se divide en tres zonas:

1. Configuración del acceso a Twitter
2. Zona de filtros de los mensajes
3. Contador de mensajes procesados

El acceso a Twitter se basa en introducir los 4 parámetros indicados en el formulario: Consumer key, Consumer secret, Token y Secret, que ya han sido explicados como se obtienen en el punto 3.5 de este documento.

Cuando el acceso esta configurado, el usuario puede filtrar los mensajes, que son las dos siguientes posibilidades:

- Seleccionar el idioma de los mensajes: es (español), en (inglés)
- Incluir términos de rastreo que deben aparecer en los tweets. Estos términos deben ir separados por comas.

Cuando la selección de filtros esta hecha, se puede proceder a hacer click en el botón “Recoger tweets”, lo cual arranca el proceso. Los mensajes que se vayan procesando con éxito son indicados en el contador: Numero de mensajes procesados.

7. Explotación de los datos

En este punto parece usar reglas de inferencia para comprobar que las instancias están correctamente creadas o que simplemente podemos obtener información de interesa sobre las mismas y sus relaciones. Así pues se activa la pestaña de SWRL en protégé.

SWRL[21] es un lenguaje propuesta para la Web Semántica que puede ser usado para expresar reglas tan bien como la lógica, combinando OWL Lite o DL con un subconjunto de reglas del lenguaje de marcado.

Una ontología OWL en la sintaxis abstracta contiene una secuencia de axiomas y hechos. Un axioma regla, o simplemente regla, se forma con un antecedente (cuerpo) y un consecuente (cabeza).

SWRL extiende el conjunto de axiomas de OWL para incluir reglas condicionales (cláusulas de Horn), del tipo: *si...entonces...*

7.1 Activación de SWRL en Protégé

Antes de empezar a crear las reglas se debe activar la pestaña en el software Protégé, así pues se deben seguir los siguientes pasos:

Seleccionar: Project | configure, que abre una ventana donde se debe marcar la casilla: SWRLTab en la pestaña de "Tab Widgets". Ahora ya aparece la pestaña: SWRL Rules, que nos permite crear reglas para nuestra ontología.

Esta nueva funcionalidad nos permite crear reglas y comprobar que su sintaxis es correcta, pero no asegura que la regla permita el resultado esperado, es decir que funcione tal y como se piensa. Así pues, adicionalmente siempre es interesante activar o incluir algún plugin que lo permita. En este proyecto se han analizado tres posibles opciones:

- SWRL-IQ[22]. Es un plugin para 3.4.x Protégé que permite a los usuarios editar , guardar y enviar consultas a un motor de inferencia subyacente basado en XSB Prolog.
- SWRLJessTab[23]. Es un plugin para Protégé que soporta la ejecución de reglas SWRL utilizando el motor de reglas Jess[24].
- SWRLDroolsTab. Es un plugin para Protégé que soporta la ejecución de reglas SWRL utilizando el motor de reglas Drools.

Los dos primeros producían inferencias en la ontología, pero el tercero no y permitía comprobar los resultados de la ejecución de las reglas sin que la ontología se viese afectada, lo cual es una característica que se tuvo en cuenta para su elección.

7.2 Creación de reglas

En primer lugar se deben aclarar que conceptos o preguntas se querían obtener mediante el uso de reglas y que no se podían obtener de otra forma. Así pues se pensó en los tres siguientes datos:

---

[21] Semantic Web Rule Language. Wikipedia. [en línea]. http://en.wikipedia.org/wiki/Semantic_Web_Rule_Language [fecha de consulta: 07 de Abril de 2014].

[22] Semantic Web Rule Language Inference and Query tool. [en línea]. http://protegewiki.stanford.edu/wiki/SWRL-IQ [fecha de consulta: 07 de Abril de 2014].

[23] SWRLJessTab. [en línea]. http://protege.cim3.net/cgi-bin/wiki.pl?SWRLJessTab [fecha de consulta: 07 de Abril de 2014].

[24] Jess. [en línea]. http://en.wikipedia.org/wiki/Jess_programming_language [fecha de consulta: 07 de Abril de 2014].

1. Obtener los Tweets más famosos del conjunto de datos.
2. Obtener los usuarios más famosos.
3. Obtener los hashtag “trending topics”.

Hay que mencionar que en todos los casos, los cálculos se basan en una suposición personal del alumno en referencia a explicaciones encontradas en Internet, y no a la realidad de Twitter, ya que en estos casos el mecanismo exacto de calculo no se conoce con exactitud.

Tweets famosos.

Al poblar la ontología parecía interesante saber que tweets eran los más relevantes del conjunto de instancias. Pero, ¿como saber si un tweet es relevante?, ya que existen varios campos de información de la clase Tweet que pueden asociarse a este concepto:

- favoriteCount. El numero de veces que el tweet ha sido marcado como favorito.
- retweetedCount. El numero de veces que el tweet ha sido retuiteado.
- User. El usuario que lo ha escrito
- etc.

En definitiva, quedaba claro que para contestar a esta pregunta iba a ser necesario consultar varias de las propiedades de un Tweet. Asi pues se propuso la siguiente definición:

“Para determinar si un Tweet es importante o “famoso”, se debe cumplir que el tweet debe haber sido marcado como favorito al menos 500 veces y haber sido retuiteado un mínimo de 1000 veces”.

Esta definición dio lugar a la siguiente regla SWRL:

tweetz:Tweet(?t) ∧ tweetz:favoriteCount(?t, ?favoritos) ∧ swrlb:greaterThan(?favoritos, 500) ∧ tweetz:retweetCount(?t, ?retuits) ∧ swrlb:greaterThan(?retuits, 1000) → sqwrl:select(?t)

En base a un Tweet: t, se obtiene el dato de la propiedad favouritesCount que se compara para saber si es mayor a 500, y además se obtiene el dato de la propiedad retweetCount que debe cumplir ser mayor a 1000, si todas las condiciones se cumplen entonces se seleccione el tweet para ser visualizado.

Usuarios famosos.

Otro factor interesante serían los usuarios que pueden ser considerados famosos. En este caso, solo parecía haber un propiedad señalable con la que interactuar para obtener este tipo de información: followersCount, que es el número de seguidores del usuario, así pues se creó la siguiente afirmación:

“Se considera un usuario famoso a aquel que dispone de más de 1000 seguidores.”

Esta definición dió lugar a la siguiente regla SWRL:

tweetz:User(?p) ∧ tweetz:followersCount(?p, ?seguidores) ∧ swrlb:greaterThan(?seguidores, 1000) → sqwrl:select(?p)

En base a un usuario: p, se obtiene el valor de la propiedad followersCount del mismo, y se compara si es mayor de 1000, en cuyo caso se selecciona para la regla.

Hashtags trending-topic.

Un trending topic (en español; recomendado, tendencia o tema del momento) es una de las palabras o frases mas repetidas en un momento concreto de Twitter. Un Trending Topic se identifica con una # (almohadilla) que preceda a uno o varios términos específicos juntos, ya

que Twitter sólo identifica como hashtag un conjunto de letras unidas. Normalmente un trending topic es un identificativo en Twitter para temas de actualidad, eventos puntuales, etc.; aunque también son palabras que en un momento dado se detectan como muy nombradas.

El algoritmo que calcula los trending topic no es público, Twitter en su página de soporte solo realiza las siguientes indicaciones al respecto[25]:

Las Tendencias se determinan mediante un algoritmo y, por defecto, se personalizan de acuerdo con las cuentas que sigues y tu ubicación geográfica. Este algoritmo identifica temas que son populares actualmente, en vez de temas que han sido populares por mucho tiempo o diariamente, para ayudarte a descubrir los temas de discusión emergentes más llamativos en Twitter más interesantes para ti.

Se tenia claro, que para disponer que temas eran más importantes que otros se tenia que averiguar con que tweets estaban relacionados y la fecha de los mismos de forma que solamente interesaban los tweets de creación reciente (menos de un mes) y si se trataba de un retweet que el tweet origen también fuese de creación reciente.

Se realizó un estudio de las funciones asociadas a propiedades de fecha, que aparecen en la página: "SWRL: A Semantic Web Rule Language[26]", pero no se encontraba la forma de comparar la propiedad: createAt de la clase Tweet con la fecha/hora actual para poder determinar si el hashtag era reciente.

Al final, solamente se pudo realizar una regla para agrupar los hashtag recopilados en el conjunto de instancias para contar el numero de veces que se repetía alguno. En base a lo cual se creó la siguiente regla:

tweetz:Hashtag(?h) ∧ tweetz:textHashtag(?h, ?name) → sqwrl:select(?name) ∧ sqwrl:count(?name)

En base a un Hashtag, h se obtiene el valor de la propiedad textHashtag del mismo, que se cuenta para determinar el numero de veces que aparece en el conjunto de instancias de la clase Hashtag.

---

[25] Twitter support: FAQs about Trends on Twitter. [en línea]. http://support.twitter.com/entries/101125 [fecha de consulta: 09 de Abril de 2014].

[26] SWRL: A Semantic Web Rule Language Combining OWL and RuleML. Built-Ins for Date, Time and Duration. [en línea]. http://www.w3.org/Submission/SWRL/#8.5 [fecha de consulta: 09 de Abril de 2014]

8. Conclusiones

Este proyecto ha supuesto el punto de partida para el aprendizaje a la Web Semántica, su estructura, lenguajes y funcionalidades. Además ha permitido el conocimiento a través de la experiencia de uso del entorno de trabajo: Protégé.

No se puede decir, que al finalizar este proyecto el alumno sea un experto en este área de conocimiento, y tampoco que éste sea el objetivo del mismo, sino simplemente obtener una base sólida sobre esta tecnología emergente.

8.1 Lecciones aprendidas

El primer punto y creo que más importante, ha sido el cambio de visión o perspectiva sobre la Web Semántica. Al principio de crear la ontología tenia un concepto asociado al proceso de crear un modelo de datos similar a una base de datos relacional, y por tanto intentaba aplicar ideas como por ejemplo la normalización[27]. Es evidente que una ontología parte en que es superior en poder expresivo a una base de datos relacional, y por ello no le son aplicables muchos de los formulismos o condicionantes que si son operativos en los modelos de bases de datos.

Otro punto a resaltar durante el proceso de investigación ha sido el número, tan alto y extenso, de aplicaciones de esta tecnología, por ejemplo al consultar la Web de: Interoperabilidad Semantica[28], el número de ontologías existentes y en tan diversos campos me resultó sorprendente; Medicina, Sistemas Geográficos, Datos Marinos, etc.

Finalmente, el dominio sobre el que se ha creado la ontología: Twitter, también ha sido una fuente de información. Se trata de un servicio que conocía pero que nunca había utilizado, porque lo asociaba mucho al servicio Facebook, pero esto ha cambiado y ahora veo que es bastante diferente. Ahora lo percibo como un servicio de divulgación de temas de actualidad muy asociados a sectores de opinión, es decir que el análisis de sus datos permite conocer tendencias y opiniones de diversos sectores de la sociedad mundial.

8.2 Objetivos

Aprender a crear una ontología sobre un área de conocimiento. Al crear una ontología sobre el servicio Web de Twitter, he tenido que investigar y utilizar (en mayor o menor grado de acierto) todos los pasos que describen el proceso de creación de una ontología OWL. El resultado obtenido como un proyecto de Protégé determina este grado de aprendizaje.

El segundo objetivo planteado, tenia como resultado: crear un programa que extraía datos de Twitter y poblará la ontología, anteriormente mencionada. El correcto funcionamiento del programa que se entrega en este proyecto certifica este segundo objetivo.

El tercer objetivo partía de la premisa de una mejora del programa en relación a la gestión de los datos que poblan la ontología. La idea inicial siempre fue incluir reglas en el programa de forma que originasen datos de interés de los individuos de la ontología, posteriormente y a media que avanzaba el proyecto, esta idea fue cambiando, principalmente por dos motivos:

a) Al principio del proyecto, mis conocimientos sobre ontologías y el dominio de la misma: Twitter, eran muy reducidos. El objetivo de la Web Semántica es permitir la creación de sistemas mucho más avanzados de gestión del conocimiento, simplificando; las ontologías se crean para dar respuesta a preguntas que otros sistemas no saben o pueden responder. Pero para hacer las preguntas hay que conocer el área de conocimiento o dominio, este no era mi caso, así pues he creado una ontología bastante “general”.
b) Al crear el programa como un plugin de Protégé, enseguida vi claro que la creación uso, y gestión de las reglas estaba ya muy bien recogido en otros plugins del sistema.

Así pues, no conseguía encontrar una forma, que aportara algo de valor, para incluir el uso de reglas en el programa, ya que estas reglas se podían crear y ejecutar desde otros plugins de forma eficaz. Finalmente, la planificación hizo que dejara de lado el uso de reglas en el programa y me centrara en la creación de las mismas desde el plugin: SWRLTab.

Por tanto, en referencia a este tercer objetivo no se si esta totalmente cumplido, porque creo que me ha faltado tiempo para profundizar en el tema de lógica e inferencias, "Reglas", sobre los datos de la ontología.

8.3 Planificación

En términos generales la planificación ha sido bastante acertada, incluso en algunos puntos del proceso como la creación de la ontología y el desarrollo básico del programa han ido con adelanto sobre el cronograma. Sin embargo, ha habido ciertos puntos que han requerido algunos cambios en el proyecto:

- En un principio se pensó en partir de una ontología existente como base para la construcción de la nuestra. El tiempo de análisis de esta tarea se pasó y me di cuenta que mi investigación únicamente había rascado en la superficie de este tema. Era más rápido no reutilizar, que seguir buscando, y si finalmente encontraba algo también requería aprender a usar dicha ontología base. Esta opción era mejor para la finalización del proyecto, pero no se si ha sido la mejor opción para crear la ontología.
-
- El objeto de las mejoras sobre el programa es donde creo se ha sufrido los cambios más importantes. De ser una "mejora del programa" ha pasado a crear reglas: SWRL para obtener información de los individuos de la ontología. Los motivos principales, están detallados en el punto anterior, pero básicamente: ambigüedad en el objetivo y tiempo.

8.4 Temas futuros.

Principalmente creo que existen mejoras en dos líneas: ontología y programa.

Ontología:

- Esta primera versión de la ontología es muy general, pero puede ser la base para crear extensiones que sirvan para responder a preguntas mas concretas. Por tanto, se puede seguir profundizando en encontrar preguntas sobre el dominio de Twitter.
- Al ser tan general, también ocurre que tenga más información de la estrictamente necesaria, por ejemplo, he recogido información sobre la altura y ancho en pixeles de las imágenes asociadas a las Tweets, ¿es realmente útil esta información?

Programa:

- Toda la información de acceso a los datos de Twitter esta dentro del programa, lo cual lo hace poco reutilizable en el caso de cambios en el Api de este servicio. Esta información podría estar fuera del código fuente, de forma configurable, para extender la reutilización del programa en versiones futuras. Este mismo cambio aplica a los nombres de las propiedades y clases de la ontología.
- Los filtros sobre los mensajes son bastante simples, y aunque Twitter no da mucho más al respecto si existía otra opción muy interesante que es filtrar por localización, pero este filtro requería introducir coordenadas geográficas y para ser útil y usable por el usuario, habría requerido integrarlo con un servicio de mapas online.

---

[27] Wikipedia. Normalización de bases de datos. [en línea]. http://es.wikipedia.org/wiki/Normalización_de_bases_de_datos [fecha de consulta: 25 de Mayo de 2014].

[28] Interoperabilidad Semántica. Ontologías. [en línea]. http://red.gnoss.com/comunidad/Interoperabilidadsemantica [fecha de consulta: 25 de Mayo de 2014].

9. Glosario

Twitter. Es un servicio de redes sociales y microblogging en línea que permite a los usuarios enviar y leer mensajes cortos de texto de un máximo de 140 caracteres, llamados “tweets”.

Tweet. Mensaje corto de texto de un máximo de 140 caracteres.

ReTweet. Es una republicación del tweet de otra persona o usuario de Twitter.

Entities. Entidades proporcionan metadatos e información contextual adicional sobre el contenido publicado en Twitter.

GeoJSON. Es un formato para la codificación de una variedad de estructuras de datos geográficos.

OWL. El Lenguaje de Ontologías Web está diseñado para ser usado en aplicaciones que necesitan procesar el contenido de la información en lugar de únicamente representar información para los humanos.

Protégé. Es un editor de ontologías, de código abierto, y un marco para la construcción de sistemas inteligentes.

10. Bibliografía

Se debe mencionar como principal referencia para el desarrollo de este proyecto, el libro: Manual de Web Semántica de Grigoris Antoniou y Fank van Harmelen (en su edición en castellano de la editorial Comares) , que ha sido fundamental para iniciar al alumno en este área.

Por otra parte, además de las menciones que se han ido indicando a lo largo del proyecto, se han consultado las siguientes Webs:

- Lenguaje de Ontologías Web (OWL) Vista General.
- Documentación Web de Twitter.
- Wiki del framework para ontologías: Protégé.
- Documentación de usuario de Protégé.
- Interoperalidad Semantica: Recursos.

11. Anexos

11.1 Análisis de integración con Twitter

A la hora de obtener información de Twitter, este servicio dispone de dos modalidades: Search Api y Stream Api. Ambas muy bien descritas en el articulo[29]: Aggregating tweets: Search vs Streaming.

En resumen se puede decir de cada una de ellas las siguientes características:

a) Stream
   - La captura de datos mediante stream se caracteriza por ser una conexión continua.
   - Las consultas pueden ser todo lo complejas que sea necesario y pueden contener varios criterios de búsqueda.
   - Los mensajes descargados vía stream son muy completos ya que contienen casi toda la información asociadas a los mismos.
   - Sólo se le permite hacer una sola conexión OAuth API de streaming para cada cuenta de twitter que posee la aplicación.

b) Search
   - La captura de datos mediante search es mucho más sencilla.
   - La obtención de datos esta influenciada por la carga del servicio.
   - El acceso viene delimitado por la cantidad de veces que se consume información.
   - Los tweets capturado contienen menos información sobre los usuarios y entidades que vía streaming.
   - Este método tiene un set de operadores más rico a la hora de hacer consultas

Hay varias características que van a propiciar que nos decantemos por una, por ejemplo:

1. Nos interesa disponer de la mayor información posible sobre los tweets a la hora de recoger la información de forma que el proceso de poblar la ontología sea lo más completo posible.
2. Aunque el programa debe poder realizar filtros, no esta planteado para disponer de una gran cantidad, sino solo de una forma para acotar el rango de registros.
3. El planteamiento del programa esta enfocado a conectar a Twitter en un determinado momento y coger un número X de tweets para poblar la ontología.

Como conclusión, parece más plausible el uso de Stream Api para conectar a Twitter.

11.2 Herramientas

Una vez conocido el tipo de conexión a realizar para obtener información de Twitter, es preciso determinar como se va a implementar la conexión, es decir que herramienta usar. La elección se debía establecer bajo las siguientes premisas:

- Como el programa debe ser un plugin del framework Protégé, es preciso programar bajo el entorno Java.
- Para acelerar el desarrollo del programa es útil usar algunas de las librerías o Apis ya implementadas para la conexión a Twitter.
- Es importante disponer de documentación y ejemplos de uso de la herramienta a utilizar

[29] 140Dev. Aggregating tweets: Search API vs. Streaming API. [en línea]. http://140dev.com/twitter-api-programming-tutorials/aggregating-tweets-search-api-vs-streaming-api/ [fecha de consulta: 09 de Mayo de 2014].

En estas circunstancias se encuentran las siguientes librerías:

a) Twitter4J[30]. Es una biblioteca Java no oficial de la API de Twitter, que permite integrar fácilmente la aplicación Java con el servicio de Twitter.
b) HoseBird[31] Client. Un cliente Java HTTP para consumir Streaming API de Twitter

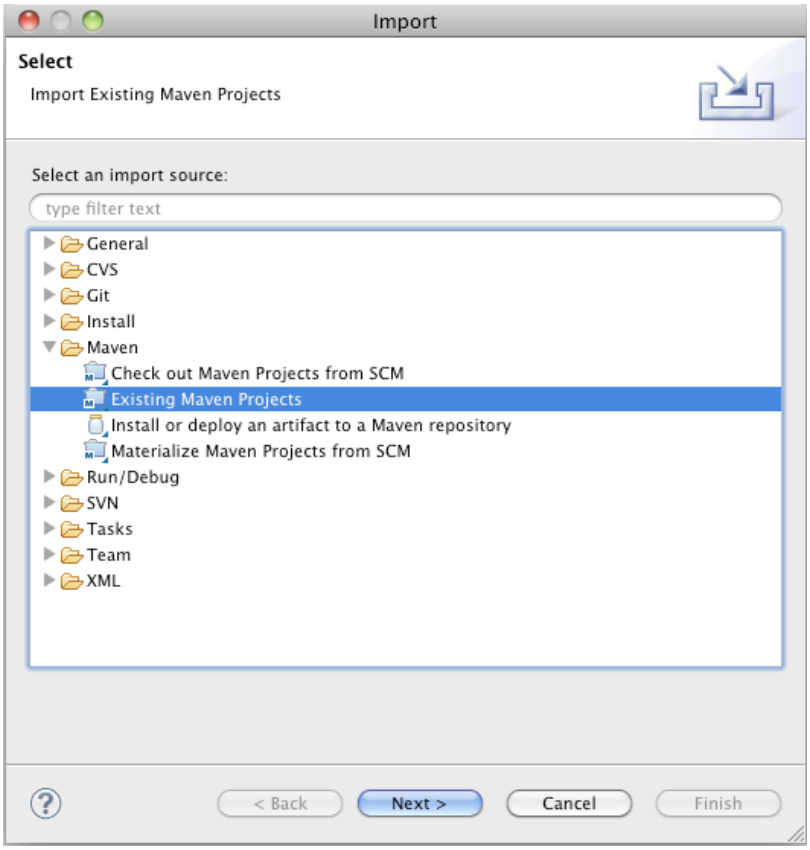
Al final la elección fue usar el cliente HoseBird porque dispone de una documentación bastante clara y unos ejemplos de uso que prácticamente funcionaron a la primera.

11.3 Compilación del programa

Para facilitar el desarrollo del programa, este ha sido creado como un proyecto maven de forma que facilitase la inclusión de las librerias auxiliares. Asi pues unicamente tendremos que utilizar la opción de importar proyecto, de tipo maven, en eclipse para empezar a utilizarlo. A continuación los detalles del proceso:

1. Debemos disponer del entorno adecuado, es decir; tener instalado una versión del framework Eclipse, y Jdk 1.6 minimo.
2. Eclipse debe tener instalado el plugin de Maven, disponible en MarketPlace, aunque normalmente ya viene instalado.
3. En este paso ya podemos importar el proyecto, asi que seleccionamos: File | Import

Al realizar esa acción se debe presentar una ventana como la siguiente, donde debemos abrir la opción Maven y hacer click en la opción: Existing Maven project



[30] Twitter4J. Official Site Home. [en línea]. http://twitter4j.org/en/index.html [fecha de consulta: 09 de Mayo de 2014].
[31] HoseBird. Github. [en línea]. https://github.com/twitter/hbc [fecha de consulta: 09 de Mayo de 2014].

4. Ahora solo debemos seleccionar la carpeta del proyecto. Eclipse inmediatamente encuentra el archivo pom.xml y sabe como importarlo de forma adecuada.
5. En este punto ya tenemos nuestro proyecto en Eclipse, el cual tiene las siguientes dependencias:
    a. protege-3.5.0. Librería core para el editor de ontologías Protégé
    b. protege-owl. Librería de ayuda que pemite el acceso al OWL modelo y sus elementos, tales como clases, propiedades o individuos.
    c. hbc-core.2.0.0. versión 2.0.0 de la librería HoseBird oficial de Twitter
    d. javax.json-api.1.0. Librería con las interfaces (Apis) del estandar JSON en Java
    e. javax.json.1.0. Librería de implementación de las interfaces anteriores.
6. Para compilar el proyecto son necesarias las librerias anteriores, algunas de las cuales no se encuentran en el repositorio central de Maven, por lo que es necesarias incluirlas[32] en nuestro repositorio local. Este proceso por linea de comandos sería de la siguiente forma:

#mvn install:install-file -DgroupId=edu.stanford.protege –DartifactId=protege –Dversion=3.5.0 –Dfile=protege-3.5.0.jar

7. Una vez el proyecto dispone de todas sus dependencias se puede compilar con la opción: clean install, creando en la carpeta target la librería:

Hay que indicar que realmente el plugin no necesita tener integrados (incluidas) las librería del protege, puesto que estas ya se encuentran en la aplicación, y por eso las excluimos del proceso de creación del fichero jar (lo cual esta indicado en el descriptor del proyecto: pom.xml), pero si son necesarias para poder compilar el fuente.

11.4 Ejemplos de mensajes en Twitter

Ejemplo 1:


```
{
	"created_at":"Wed Mar 19 18:41:49 +0000 2014",
	"id":446355881329168384,
	"id_str":"446355881329168384",
	"text":"As\u00ed que se\u00f1orita si a usted le molesta que me ponga al lado suyo para aprovechar la sombra, caguese. No me pienso cagar de calor por usted",
	"source":"\u003ca href=\"https:\/\/twitter.com\/download\/android\" rel=\"nofollow\"\u003eTwitter for Android\u003c\/a\u003e",
	"truncated":false,
	"in_reply_to_status_id":null,
	"in_reply_to_status_id_str":null,
	"in_reply_to_user_id":null,
	"in_reply_to_user_id_str":null,
	"in_reply_to_screen_name":null,
	"user":{
		"id":227839638,
		"id_str":"227839638",
		"name":"El imprescindible",
		"screen_name":"valengarione",
		"location":"C\u00f3rdoba, Argentina",
		"url":"https:\/\/www.facebook.com\/vale.garione",
		"description":"El que ama, sufre\r\nEl que sufre, lucha\r\nEl que lucha, gana",
		"protected":false,
		"followers_count":269,
		"friends_count":268,
		"listed_count":0,
```


[32] Apache Maven project. Usage maven install plugin. [en línea]. http://maven.apache.org/plugins/maven-install-plugin/usage.html [fecha de consulta: 15 de Mayo de 2014].

```
                        "created_at":"Fri Dec 17 23:45:12 +0000 2010",
                        "favourites_count":215,
                        "utc_offset":-10800,
                        "time_zone":"Buenos Aires",
                        "geo_enabled":true,
                        "verified":false,
                        "statuses_count":10512,
                        "lang":"es",
                        "contributors_enabled":false,
                        "is_translator":false,
                        "is_translation_enabled":false,
                        "profile_background_color":"0099B9",
            "profile_background_image_url":"http:\/\/pbs.twimg.com\/profile_background_images\/700971243\/942bc9f6b0
28b8370c8a006cc3664f1d.jpeg",

            "profile_background_image_url_https":"https:\/\/pbs.twimg.com\/profile_background_images\/700971243\/942
bc9f6b028b8370c8a006cc3664f1d.jpeg",
                        "profile_background_tile":true,

            "profile_image_url":"http:\/\/pbs.twimg.com\/profile_images\/442773028842127360\/UKDGVGyG_normal.jpeg"
,

            "profile_image_url_https":"https:\/\/pbs.twimg.com\/profile_images\/442773028842127360\/UKDGVGyG_norm
al.jpeg",
                        "profile_banner_url":"https:\/\/pbs.twimg.com\/profile_banners\/227839638\/1389904442",
                        "profile_link_color":"B80019",
                        "profile_sidebar_border_color":"FFFFFF",
                        "profile_sidebar_fill_color":"95E8EC",
                        "profile_text_color":"3C3940",
                        "profile_use_background_image":true,
                        "default_profile":false,
                        "default_profile_image":false,
                        "following":null,
                        "follow_request_sent":null,
                        "notifications":null
            },
            "geo":null,
            "coordinates":null,
            "place":null,
            "contributors":null,
            "retweet_count":0,
            "favorite_count":0,
            "entities":{
                        "hashtags":[

                        ],
                        "symbols":[

                        ],
                        "urls":[

                        ],
                        "user_mentions":[

                        ]
            },
            "favorited":false,
            "retweeted":false,
            "filter_level":"medium",
            "lang":"es"
}
```


Este mensaje consta de las propiedades básicas de un tweet, y su usuario creador.

Ejemplo 2:


```
{
            "created_at":"Mon Mar 24 18:50:13 +0000 2014",
            "id":448169934166982656,
            "id_str":"448169934166982656",
            "text":"RT @Ovaciondigital: Su\u00e1rez es l\u00edder en la carrera por la Bota de Oro. Lleva 28 goles, dos
m\u00e1s que Ronaldo. Repas\u00e1 sus mejores tantos http:\/\u2026",
            "source":"web",
```

```
"truncated":false,
"in_reply_to_status_id":null,
"in_reply_to_status_id_str":null,
"in_reply_to_user_id":null,
"in_reply_to_user_id_str":null,
"in_reply_to_screen_name":null,
"user":{
"id":1435296374,
"id_str":"1435296374",
"name":"paquete ",
"screen_name":"alfredo790213",
"location":"",
"url":null,
"description":"Nadie es tan rico que no necesita una sonrisa. \nNadie es tan pobre que no pueda
darla olimpia, mi unica piel!!!!",
"protected":false,
"followers_count":516,
"friends_count":593,
"listed_count":0,
"created_at":"Fri May 17 10:32:25 +0000 2013",
"favourites_count":110,
"utc_offset":-10800,
"time_zone":"Atlantic Time (Canada)",
"geo_enabled":true,
"verified":false,
"statuses_count":6300,
"lang":"es",
"contributors_enabled":false,
"is_translator":false,
"is_translation_enabled":false,
"profile_background_color":"C0DEED",

"profile_background_image_url":"http:\/\/pbs.twimg.com\/profile_background_images\/378800000027793965\/
e79e360def232bd82de738a4e98c84c5.jpeg",

"profile_background_image_url_https":"https:\/\/pbs.twimg.com\/profile_background_images\/3788000000277
93965\/e79e360def232bd82de738a4e98c84c5.jpeg",
"profile_background_tile":false,

"profile_image_url":"http:\/\/pbs.twimg.com\/profile_images\/440442175927500800\/25kGFd_l_normal.jpeg",

"profile_image_url_https":"https:\/\/pbs.twimg.com\/profile_images\/440442175927500800\/25kGFd_l_normal.j
peg",
"profile_banner_url":"https:\/\/pbs.twimg.com\/profile_banners\/1435296374\/1372852606",
"profile_link_color":"0084B4",
"profile_sidebar_border_color":"FFFFFF",
"profile_sidebar_fill_color":"DDEEF6",
"profile_text_color":"333333",
"profile_use_background_image":true,
"default_profile":false,
"default_profile_image":false,
"following":null,
"follow_request_sent":null,
"notifications":null
},
"geo":null,
"coordinates":null,
"place":null,
"contributors":null,
"retweeted_status":{
"created_at":"Mon Mar 24 12:13:24 +0000 2014",
"id":448070072766967808,
"id_str":"448070072766967808",
"text":"Su\u00e1rez es l\u00edder en la carrera por la Bota de Oro. Lleva 28 goles, dos m\u00e1s
que Ronaldo. Repas\u00e1 sus mejores tantos http:\/\/t.co\/QuS6pf4AkG",
"source":"\u003ca href=\"http:\/\/twitter.com\/tweetbutton\" rel=\"nofollow\"\u003eTweet
Button\u003c\/a\u003e",
"truncated":false,
"in_reply_to_status_id":null,
"in_reply_to_status_id_str":null,
"in_reply_to_user_id":null,
"in_reply_to_user_id_str":null,
"in_reply_to_screen_name":null,
"user":{
"id":146919703,
```

```
                                "id_str":"146919703",
                                "name":"Ovaci\u00f3n Digital",
                                "screen_name":"Ovaciondigital",
                                "location":"Montevideo",
                                "url":"http:\/\/www.ovaciondigital.com.uy",
                                "description":null,
                                "protected":false,
                                "followers_count":67369,
                                "friends_count":12,
                                "listed_count":227,
                                "created_at":"Sat May 22 18:00:09 +0000 2010",
                                "favourites_count":12,
                                "utc_offset":-10800,
                                "time_zone":"Buenos Aires",
                                "geo_enabled":true,
                                "verified":false,
                                "statuses_count":35815,
                                "lang":"es",
                                "contributors_enabled":false,
                                "is_translator":false,
                                "is_translation_enabled":false,
                                "profile_background_color":"137D1A",

        "profile_background_image_url":"http:\/\/pbs.twimg.com\/profile_background_images\/382487538\/fondo_twitter_ovacion.jpg",

        "profile_background_image_url_https":"https:\/\/pbs.twimg.com\/profile_background_images\/382487538\/fondo_twitter_ovacion.jpg",
                                "profile_background_tile":true,

        "profile_image_url":"http:\/\/pbs.twimg.com\/profile_images\/378800000410371114\/254afc39ef0b2bc6c1e5dd64506c5058_normal.jpeg",

        "profile_image_url_https":"https:\/\/pbs.twimg.com\/profile_images\/378800000410371114\/254afc39ef0b2bc6c1e5dd64506c5058_normal.jpeg",
                                "profile_link_color":"095C05",
                                "profile_sidebar_border_color":"EEEEEE",
                                "profile_sidebar_fill_color":"EFEFEF",
                                "profile_text_color":"0F0F0F",
                                "profile_use_background_image":true,
                                "default_profile":false,
                                "default_profile_image":false,
                                "following":null,
                                "follow_request_sent":null,
                                "notifications":null
                        },
                        "geo":null,
                        "coordinates":null,
                        "place":null,
                        "contributors":null,
                        "retweet_count":13,
                        "favorite_count":6,
                        "entities":{
                                "hashtags":[

                                ],
                                "symbols":[

                                ],
                                "urls":[
                                        {
                                                "url":"http:\/\/t.co\/QuS6pf4AkG",
                                                "expanded_url":"http:\/\/www.ovaciondigital.com.uy\/futbol\/suarez-bota-oro.html",
                                                "display_url":"ovaciondigital.com.uy\/futbol\/suarez-\u2026",
                                                "indices":[
                                                        113,
                                                        135
                                                ]
                                        }
                                ],
                                "user_mentions":[

                                ]
                        },
```

```
                "favorited":false,
                "retweeted":false,
                "possibly_sensitive":false,
                "lang":"es"
        },
        "retweet_count":0,
        "favorite_count":0,
        "entities":{
                "hashtags":[

                ],
                "symbols":[

                ],
                "urls":[
                        {
                                "url":"http:\/\/t.co\/QuS6pf4AkG",
                                "expanded_url":"http:\/\/www.ovaciondigital.com.uy\/futbol\/suarez-bota-
oro.html",
                                "display_url":"ovaciondigital.com.uy\/futbol\/suarez-\u2026",
                                "indices":[
                                        139,
                                        140
                                ]
                        }
                ],
                "user_mentions":[
                        {
                                "screen_name":"Ovaciondigital",
                                "name":"Ovaci\u00f3n Digital",
                                "id":146919703,
                                "id_str":"146919703",
                                "indices":[
                                        3,
                                        18
                                ]
                        }
                ]
        },
        "favorited":false,
        "retweeted":false,
        "possibly_sensitive":false,
        "filter_level":"medium",
        "lang":"es"
}
```


En este segundo ejemplo, el tweet tiene asociado otro tweet porque se trata de un retweet, como vemos en la propiedad: retweeted_status. Además dispone de entidades, y en definitiva contiene mucha más información.