InfoPython - Midiendo el Valor de la Información de Mass Media con Python

	**Autor:** Juan Bautista Cabral JBC conoció python una solitaria noche del 2007. Desarrolló su proyecto de grado de la carrera Ingeniería en Sistemas con este lenguaje utilizando el framework Django y trabajó 1 año desarrollando evaluadores de información usando nuestro querido reptil. **twitter:** @leliel12 **blog:** http://jbcabral.wordpress.com **mail:** jbc.develop@gmail.com

Infopython es una librería para la valoración de medios de información que utiliza teorías formales provenientes de las ciencias sociales. Inicialmente, eran un conjunto de módulos dispersos que utilizaba en mi trabajo; luego de unos días de refactoring y paciencia logré unificarlos en una única librería.

Trasfondo

Existen diferentes teorías sociológicas para determinar la importancia de los medios sobre la opinión pública, las mismas analizan la información que estos emiten desde el punto de vista del emisor, el receptor o ambos.

Por citar un ejemplo, la **Teoría de Información de Shannon** es una formalización matemática de una de una teoría sociológica conocida como la **Aguja Hipodérmica**.

En el caso de **Infopython** la teoría que nos ocupa es conocida como **Agenda-Setting** (en el futuro se planean agregar otras); la cual postula que los medios de comunicación de masas tienen una gran influencia sobre el público al determinar qué historias poseen interés informativo, y cuánto espacio e importancia se les da. El punto central de esta teoría es asignar una prioridad para obtener mayor **audiencia**, mayor **impacto** y una determinada **conciencia** sobre la noticia.

Los Medios

En **Infopython**, antes de mencionar **como** son los procesos de cálculo del valor de la información es necesario determinar **qué** es lo que vamos a medir.

Así definimos informalmente que para nuestro dominio un medio de información es:

> **Un elemento emisor sobre el cual quiero efectuar una medición del valor de su información. Esta Información tiene como características: ser homogénea, dar una "sensación" de unidad, y ser medible.**

Siendo:

- **Homogénea:** Toda información que emite el medio tiene que tener características comunes. Sino dada su extrema variación interna el medio se nos hace imposible de medir.

  Por Ejemplo:

  un canal de televisión, para nuestro caso, no califica como "medio" ya que cada programa y franja horaria tiene **muy** diferentes niveles de audiencia y contenido; en este caso es mejor tomar como unidad cada programa de televisión como el medio a medir.

- **Sensación de Unidad:** Es más fácil entender este concepto por medio de un ejemplo:

  Si yo digo que mi medio de información son "Revistas de Deporte" da una sensación de que este elemento no es "un solo medio", pero sin embargo cuando cambio la definición

del medio, a "La revista de deportes Goles y Faroles" esta sensación sí esta presente.

- **Medible:** Si de un medio que definimos no podemos extraer datos cuantitativos, no tiene sentido para nosotros.

Formalizando

Llegado el punto en el cual ya tenemos definidos **que es un medio** para nuestro dominio, podemos definir "matemáticamente" un modelo que se ajuste a la definición previa de la "Agenda-Setting"; así proponemos lo siguiente:

> **El Valor de la información de un medio es una función de la audiencia y el impacto, descartando la conciencia ya que es difícil o imposible de medir**

O lo que es lo mismo:

> **VALOR = F(AUDIENCIA, IMPACTO)**

Siendo:

- **VALOR:** Es la importancia del medio dada la teoría.
- **AUDIENCIA:** A cuanta gente le llega la información del medio.
- **IMPACTO:** Qué tanta importancia le da la audiencia al medio.

Ahora como ultimo se propone como **F** a:

> **F(AUDIENCIA, IMPACTO) = AUDIENCIA * IMPACTO**

Se elige la función **'*'** (Multiplicación) por los siguientes motivos:

- **Refleja mejor la variación de los valores:** si un parámetro crece o decrece mucho, hace variar mucho al valor.

  Supongamos el siguiente caso:

  > Un medio el cual es seguido por un número bajo de audiencia **10** pero tiene un alto impacto **1.000**. Esto puede darse si esas pocas personas pertenecen a un grupo de influencia (asesores presidenciales por ejemplo).
  >
  > En este caso el valor de la información sería **10.000**, y dado qué valor definimos como "mucho" y qué valor definimos como "poco", este valor es "grande" (mucho). Esto, lo podemos considerar como correcto, ya que cualquier medio que pueda influir en personas importantes debería tener un alto valor.
  >
  > Manteniendo los valores, pero cambiando nuestra función por **AUDIENCIA + IMPACTO** el valor de la información sería **1.010** el cual, manteniendo el razonamiento anterior sigue siendo alto.
  >
  > Ahora, si reemplazamos el valor de la audiencia por un número grande **1000**, y mantenemos el del impacto; el valor de la función original sería **1 millón** y en el segundo caso **2.000**.
  >
  > Si pensamos que ahora impactamos probablemente sobre 1000 asesores presidenciales el valor **2.000** se queda chico, ya que estamos en presencia de un medio que probablemente genere un impacto a nivel global. Lo cual evidencia que la multiplicación representa mucho mejor la variación de parámetros.

- **El valor se anula ante la ausencia de audiencia o de impacto**: Cuando la audiencia o el impacto son **0 (cero)** (Nadie ve o nadie presta atención al medio) el valor de la información también es **0 (cero)**.

  Esto no es trivial ya que nos sugiere que la información no vale nada si nadie le interesa verla o nadie le presta atención.

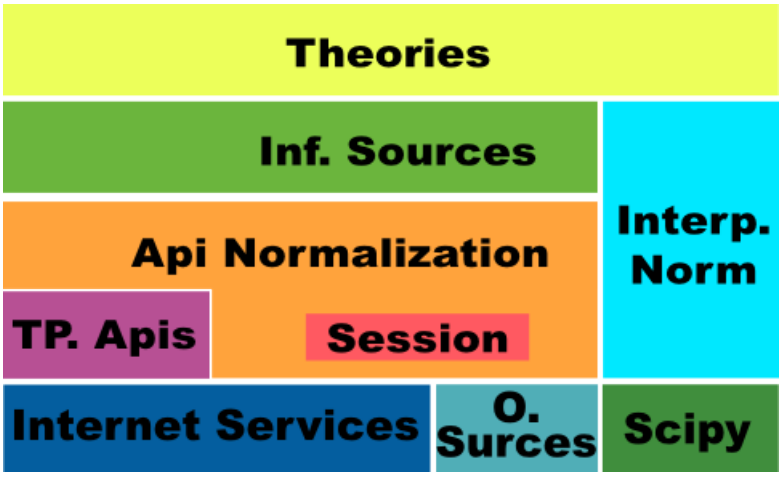
Infopython

Dado que existe una amplia variedad de servicios públicos que extraen estadísticas y datos sobre nuevos medios (web, twitter, etc), como por ejemplo:

- Klout (http://klout.com/)
- Compete (http://www.compete.com/)
- Alexa (http://www.alexa.com/)

por citar algunos; **Infopython** se centra en brindar un API sencilla para valorar a través de agenda-setting (en el futuro se implementarán otras teorías)a los medios independientemente de su tipo, utilizando los servicios antes mencionados

Arquitectura:



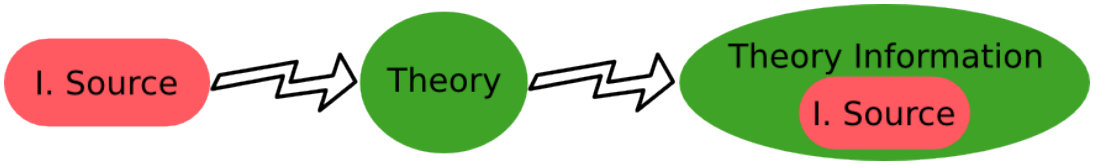
Analizamos Cada Capa:

- **Internet Service:** Corresponde a los distintos servicios que existen en la web para la extracción de estadísticas y datos de los nuevos medios.
- **Other Sources:** Son otros datos que con los que se alimenta a **Infopython**, como ser Bases de datos, plantillas excel, etc.
- **Scipy:** Es una biblioteca de código abierto de algoritmos y herramientas matemáticas.

  Esta se encarga del procesamiento numérico necesario.
- **Third Parties Apis:** Son librerías de terceros que se conectan a servicios que existen en la red. Por ejemplo:

  - tweepy que sirve para manipular datos de twitter.
  - koutpy que se conecta a Klout
- **Session:** Esta sub-capa es un módulo que se encarga de centralizar todas las configuraciones necesarias para acceder a los servicios de internet.
- **Interpolation Normalization:** Esta es una capa de abstracción para los diferentes interpoladores que posee Scipy y define algunos nuevos, todos con la misma API.
- **API Normalization:** Se encarga de convertir todas las respuestas de todos los servicios de internet y las API's de terceros a estructuras comunes (diccionarios) utilizando de ser necesarios los datos que posee la session.

- **Information Sources:** Son las clases que representan nuestras fuentes de información. Las mismas están conectadas de manera "auto-mágica" a las diferentes API's Normalizadas.
- **Theories:** Esta capa posee módulos que definen el comportamiento y los cálculos de las teorías implementadas en la **Infopython** (para la versión actual solo Agenda-Setting). Cada teoría encapsula los medios de de información en "nodos" los cuales agregan los datos que brinda dicha teoría.



Ahora definida toda la teoría, y toda la arquitectura, podemos mencionar cómo se trabaja con la librería:

1. **Configurar la sesión:** Consiste en brindarle a la capa de sesión todas las api key (mecanismos de autentificación de servicios de tercero) que requiera.

   Ejemplo:

```
from infopython import session

# Listado de todas las llaves OBLIGATORIAS de la librería
session.NEEDED_KEYS

# configura la session con las llaves v0, v1, ...
session.set(v0=1, v1=2...)

# retorna el valor de una llave
session.get("v0")

# borra la session
session.clear()
```

   En la versión actual todas las NEEDED_KEY son obligatorias y la sesión es inmutable.

2. **Crear los medios: Crear los medios de información sobre los cuales** se desea consultar su valor. En esta versión de **Infopython** se brinda clases para **2** medios:

   - `WebPages:` Representa una página web independientemente si ésta es un perfil de twitter o un blog, o lo que fuera. Se sugiere como mecanismo de medición de audiencia los servicios de Compete (http://www.compete.com/) o los de Alexa (http://www.alexa.com/).

     Y como mecanismo de medición de impacto Page Rank (http://es.wikipedia.org/wiki/PageRank), ya que si Google dice que la importancia de información es ésta, no vamos a discutir con Google.

     Ejemplo del Api de `WebPage`:

```
from infopython.isources import webpages

google = webpages.WebPage("google.com")

google.id # devolveria "google.com"
google.url # devolveria "http://google.com"
google.html # El contenido en HTML de "http://google.com"
google.text # El texto del HTML de "http://google.com"

google.get_info("compete") # la informacion de compete de
                           # "google.com" utilizando el
                           # key de compete suministrado
                           # en la session
```

   - `TwitterUser:` Representa un usuario de Twitter y NO sus tweets

     Se sugiere como mecanismo de medición de audiencia la cantidad de followers; y de impacto la información suministrada por Klout (http://klout.com/)

     Ejemplo del Api de `TwitterUser`:

```
from infopython.isources import twitteruser

yo = twitteruser.TwitterUser("leliel12")
yo.id # leliel12
yo.username # leliel12
yo.get_info("tweepy") # la informacion de tweepy del usuario
                      # "leliel12" utilizando el key de
                      # Twitter suministrado en la session
```

3. **Crear Evaluadores:** Consiste en crear **callables** (funciones o métodos) que reciban un medio de información como parámetro y devuelvan los valores que se asumirán como audiencia o impacto. Por ejemplo si decidimos que nuestra isource `WebPage` extraerá su **audiencia** de **Compete** y su **Impacto** de **Pagerank**, la funciones deberían ser similares a estas:

```
# extrae los unique visitors de compete de la WebPage que recibe como
# parámetro
aud = lambda w: w.get_info("compete")["metrics"]["uv_count"]

# Extrae el valor de page rank de la WebPage que recibe como parámetro
imp = lambda w: w.get_info("pagerank")["pagerank"]
```

Si a la agenda no le sumistramos alguno de los evaluadores, ésta tratará de usar los interpoladores suministrados.

4. **Crear los interpoladores:** Los interpoladores se utilizan como segunda alternativa a la extracción de **audiencia** e **impacto**, por lo que cada agenda recibe 2 interpoladores: un interpolador de audiencia y uno de impacto.

   Así el interpolador de **impacto** recibirá como valor para interpolar **"X"** a la **audiencia** y devolverá un valor **"Y"** correspondiente al **impacto***.

   Ahora, si lo que deseamos es interpolar el valor de la **Audiencia**, el interpolador recibirá como valor **"X"** el **Impacto** y devolverá un valor **"Y"** correspondiente a la **Audiencia**.

   Se mostrará un ejemplo en conjunto más adelante.

5. **Crear la/s agenda/s:** Al crear las agendas se les debe suministrar diferentes datos:

   - Qué tipo de medio de información medirá.
   - Una lista de medios de información a medir(opcional).
   - Un extractor de datos de audiencia (opcional).
   - Un extractor de datos de impacto (opcional).
   - Un interpolador de audiencia (opcional).
   - Un interpolador de impacto (opcional).

   Se mostrará un ejemplo en conjunto más adelante.

6. **Evaluar los nodos:** La agenda posee métodos para ordenar los `ISources` según su valor, para luego ser iterada y así generar un ranking de importancia de cada medio.

   Al iterar sobre la `Agenda`, ésta devuelve varios `ASNode` los cuales son estructuras de datos que encapsulan a los medios y agregan atributos correspondientes a **Audiencia**, **Impacto** y **Valor** así como también fecha y hora de cuando fue creado el nodo.

Ejemplo Completo

```
from infopython import session
from infopython import agenda
from infopython.util import interpolator
from infopython.isources import webpages

# Configuramos la session.
# Todas estas llaves son de fantasía y para una prueba real cualquier
# Usuario puede registrarlas en la pagina de cada aplicación.
session.set(compete_key = "967b8490-e26a-11df-8cbe-0019662306b1",
            twitter_key = "967b8490-e26a-11df-8cbe-0019662306b1",
            twitter_secret = "967b8490-e26a-11df-8cbe-0019662306b1",
            twitter_user_key = "967b8490-e26a-11df-8cbe-0019662306b1",
            twitter_user_secret = "967b8490-e26a-11df-8cbe-0019662306b1",
            klout_api_key = "967b8490-e26a-11df-8cbe-0019662306b1")

# Creamos dos webpages
google = webpages.WebPage("google.com")
yahoo = webpages.WebPage("yahoo.com")

# Saca cosos
aud = lambda w: w.get_info("compete")["metrics"]["uv_count"] # audiencia
imp = lambda w: w.get_info("pagerank")["pagerank"] # impacto

# un interpolador
itp = interpolator.PieceWisePolynomial([0,0,1,1,2,45,64], [1,3,1,1,2,4,64])

# Creamos la agenda
# Esta agenda tratara de extraer los valores de audiencia e impacto con su
# 'valuators',  en caso de volver 'None' lo intentará con sus interpoladores.
# Si estos vuelven a devolver None, se retornará como valor 0.0 y se calculará
# el valor del medio con ellos.
ag = agenda.AgendaSetting(itype=webpages.WebPage,
                          inf_sources=[google, yahoo],
                          audience_valuator=aud,
                          impact_valuator=imp,
                          impact_interpolator=itp,
                          audience_interpolator=itp)

ag.rank() # ordenamos la agenda por el valor de cada medio

# Iteramos sobre cada ASNode e imprimimos los valores de audiencia e impacto.
for i in ag:
    print i.id, "%s + %s = %s" % (i.audience, i.impact, i.value)
```

Más Métodos de la Agenda

Suponiendo que tenemos una instancia, la misma agenda del ejemplo anterior `ag` y el `WebPage`, `google`:

```
ag.value_of(google) # devuelve el valor de google (audiencia + impacto)
ag.impact_of(google) # devuelve el valor del impacto de google
                     # o sea dado lo que definimos como evaluador de
                     # impacto haría la llamada:
```

```
                    # return google.get_info("pagerank")["pagerank"]

ag.audience_of(google) # devuelve el el valor de la audiencia de google
                       # osea dado lo que definimos como evaluador de audiencia
                       # haría la llamada:
                       # return google.get_info("compete")["metrics"]["uv_count"]

ag.wrap(google) # Devolvería un ASNode con los valores de audiencia,
                # impacto y valor de la información de google

ag.count(google) # Devuelve cuantas veces aparece este medio en la agenda

ag.remove(google) # elimina la primer ocurrencia google en la agenda

ag.append(google) # agrega google a la agenda

ag.for_type # Devolveria para que tipo de isource fue creada esta agenda
            # WebPage para nuestro ejemplo

ag.audience_valuator # None o la función de calculo de audiencia

ag.impact_valuator # None o la función de calculo de impacto

ag.audience_interpolator # None o el interpolador de audiencia

ag.impact_interpolator # None o el interpolador de impacto
```

Comparando 2 Agendas

En el módulo `agenda` existe una función que es muy útil para evaluar varias agendas con diferentes medios de información.

Esta función retorna una lista de `ASNode` ordenada de ambas agendas.

```
from infopython import agenda
from infopython.isources import webpages, twitteruser

# 2 agendas con diferentes tipos de medios.
ag1 = agenda.AgendaSetting(isource=webpages.WebPage)
ag2 = agenda.AgendaSetting(isource=twitteruser.TwitterUser)

# itera sobre todos los medios de informacion de ambas agendas
# ordenados por 'value'.
for i in agenda.rank_isources(ag1, ag2):
    print i
```

Nota Final: Test

Al bajar la librería lo primero que debe hacerse es correr el test con los siguientes pasos:

1. **Correr**

   ```
   $ python setup.py test
   ```

2. Configurar `test.cfg` con las llaves de las API's correspondientes.

3. **Correr ahora si**

   ```
   $ python setup.py test
   ```

Conclusión

Como vimos **Infopython** provee una manera uniforme para la valoración de la información. En versiones futuras se planea introducir otros tipos de mass-media ya que por ejemplo,**IMDB** y **GoogleBooks** provee información vía API's de medios tradicionales (películas y libros); o, yendo mas allá, **LinkedIn** información bastante confiable de perfiles laborales.

También es posible la integración con el procesamiento de lenguaje natural con NLTK o alguna herramienta de la web semántica.

Enlaces:

- Infopython: http://bitbucket.org/leliel12/infopython/
- Teoría de Agenda-Setting: http://en.wikipedia.org/wiki/Agenda-setting_theory