A semantic news aggregator in Python using Dbpedia, Cubicweb and Scikits-learn

Vincent Michel - Logilab

26 août 2011

Overview

Context

With a bunch of RSS news...

- *Google Borrows Apple Strategy : Google's deal underscores the allure of a business model pioneered...*
- *Japan disaster plant cold shutdown could face delay : TOKYO (Reuters) - Tokyo Electric Power Co said on Wednesday...*
- *Libya shows signs of slipping from Muammar Gaddafi's grasp : Supply lines to capital in peril as coastal cities fall...*

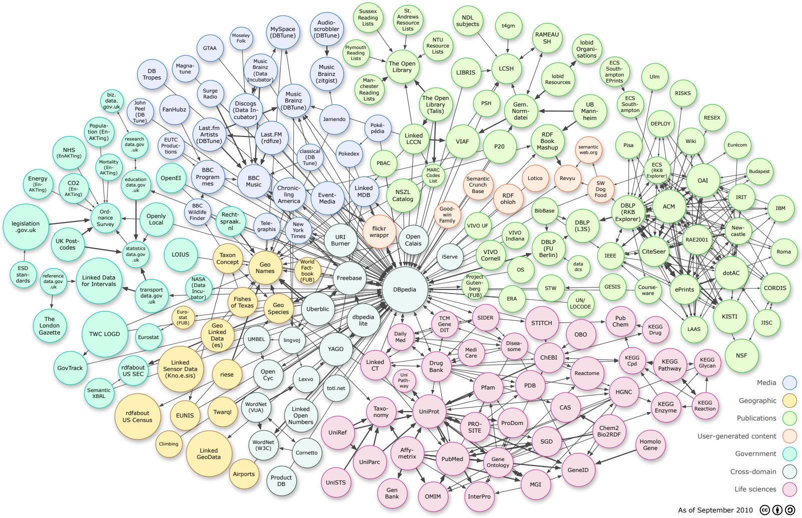
... how can we analyze them in Python ?

$\rightarrow$ Clustering (grouping) RSS (*e.g. Google News*).

$\rightarrow$ **Extracting/synthetizing information.**

$\rightarrow$ **Providing useful/original visualisation and analytics tools.**

Semantic ?



"Linking Open Data cloud diagram, by Richard Cyganiak and Anja Jentzsch.
http ://lod-cloud.net/"

Tools

Fetching, storing and querying the data $\rightarrow$ CubicWeb (*)

- ✓ Semantic CMS, high-level database management with metadata.
- ✓ Multiple sources : RSS, micro-blogging, SQL database, . . .
- ✓ Based on **PostgreSQL** : deals with (very) large database.

Data-mining and machine learning $\rightarrow$ Scikits-learn (*)

- ✓ Easy-to-use and general-purpose machine learning in Python.
- ✓ Unsupervised learning, supervised learning, model selection . . .

Semantic information database $\rightarrow$ Dbpedia (*)

- ✓ $\sim 8.10^6$ articles with *abstracts, images, . . .* from Wikipedia.
- ✓ $\sim 0,6.10^6$ categories, 273 types (*e.g. person, place, . . .*)
- ✓ $\sim 100.10^6$ links between articles, categories and types.

(*) Open Source/Creative Commons ! ! !

Storing RSS information with **Cubicweb**

Each object (or **entity**) is defined in a **schema** and may be displayed using different **views**.

Storing RSS

Define (in *schema.py*) how RSS should be stored in the database.

```
class RSSArticle(EntityType):
    title = String() # Title of the feed
    uri = String(unique=True) # Uri of the rss feed
    content = String() # Content of the feed
```

Fetching RSS (based on *feedparser* and *BeautifulSoup*)

Simply construct a **source**, by giving an **URL** and a **parser** :

```
url = u'http://feeds.bbci.co.uk/news/world/rss.xml'
s = session.create_entity('CWSource', name=u'BBCNews-World', url=url,
                          type=u'datafeed', parser=u'rss-parser',
                          config=u'synchronization-interval=240min')
s.pull_data(session)
```

7 english/american journals (*The New York Times*, . . .)

Storing Dbpedia information with **Cubicweb**

Storing Dbpedia page

Define (in *schema.py*) how Dbpedia pages should be stored in the database.

```
class DbpediaPage(EntityType):
    uri = String(unique=True, indexed=True) # Uri of the ressource
    label = String(indexed=True) #http://www.w3.org/2000/01/rdf-schema
    pageid = String() # http://dbpedia.org/ontology/wikiPageID
    abstract = String() # http://dbpedia.org/ontology/abstract
    homepage = String() # http://xmlns.com/foaf/0.1/homepage
    thumbnail = String() # http://dbpedia.org/ontology/thumbnail
    depiction = String() # http://xmlns.com/foaf/0.1/depiction
    wikipage = String() # http://xmlns.com/foaf/0.1/page
    latitude = String() # http://www.w3.org/2003/01/geo/wgs84_pos
    longitude = String() # http://www.w3.org/2003/01/geo/wgs84_pos
```

Storing all dbpedia information ($\sim 9.10^6$ pages, $\sim 100.10^6$ links, $20\,Go$) in Cubicweb takes **less than 24 hours.**

$\rightarrow$ See the full schema

Analyzing RSS news

What can we do with the RSS news stored in database ?

1. extract **relevant features of the data**.
2. construct a **usable (i.e. matrix) representation of the data**.
3. **clusters** (group) RSS together.
4. deeper **analyze/visualization** of the information.

Example sentence :

“Google is to buy mobile phone manufacturer Motorola Mobility, allowing it to mount a serious challenge to Apple Inc.”

Overview

Extracting information : Classical approaches (**Scikits-learn**)

Char-N-gram

Extracts tokens of *N* characters from a text.

```
from scikits.learn.feature_extraction.text import CharNGramAnalyzer
analyzer = CharNGramAnalyzer(min_n=3, max_n=6)
tokens = analyzer.analyze(sentence)
```

450 tokens : *'goo', 'oog', 'ogl', . . . , 'mob', 'obi', 'bil', . . .*

Word-N-gram

Extracts tokens of *N* words from a text.

```
from scikits.learn.feature_extraction.text import WordNGramAnalyzer
analyzer = WordNGramAnalyzer(min_n=3, max_n=6)
tokens = analyzer.analyze(sentence)
```

58 tokens : *'google is to', 'is to buy', . . . , 'serious challenge to', . . .*

- ✘ Many irrelevant features.
- ✘ Features do not carry lots of contextual information (i.e. **understandable by humans**).

Feature extraction : Dbpedia - Context

Main Hypothesis : Only things that exist in Dbpedia (*i.e* Wikipedia) have some interest in news analysis $\rightarrow$ Named Entities Recognition (NER)

Dbpedia feature extraction (**Cubicweb/Dbpedia**)

```
from cubes.semnews.views.nertools import DbpediaEntitiesAnalyzer
analyzer = DbpediaEntitiesAnalyzer(session, lang='en')
tokens = analyzer.analyze(sentence)
```

$\rightarrow$ 3 tokens : *'Apple Inc', 'Google', 'Motorola'*

<ENAMEX TYPE="ORGANIZATION"> Google</ENAMEX> is to buy mobile phone manufacturer <ENAMEX TYPE="ORGANIZATION"> Motorola</ENAMEX> Mobility, allowing it to mount a serious challenge to <ENAMEX TYPE="ORGANIZATION"> Apple Inc</ENAMEX>

$\rightarrow$ Try it !

"DBpedia Spotlight : Shedding Light on the Web of Documents", Pablo N. Mendes et al., I-Semantics 2011
"Learning Named Entity Recognition from Wikipedia", Joel Nothman 2008
"Large-Scale Named Entity Disambiguation Based on Wikipedia Data", Silviu Cucerzan 2007

Feature extraction : Dbpedia - Properties

Efficient and robust feature extraction

- **Keep the meaning of a text** $\rightarrow$ interpretable features.
  - ✗ *'. . . said former soldier Larry, . . . '*
  - ✗ *'. . . said student Larry, . . . '*
  - ✓ *'. . . said Larry Page, . . . '*
- **Robust features based on redirections.**

  e.g. 'Obama', 'Barak Obamba', 'Pres. Obama' redirects to *'Barack Obama'*

Simple RQL (*Relation Query Language*) queries

- **Fast**, based on indexed SQL tables and regular expressions :

```
rset = rql('Any E WHERE E is DbpediaPage,
                             E label %(token)s', {'token': token})
```

  e.g. 19 entities extracted in 4s in 765 words, among $\sim 8.10^6$ dbpedia entries.

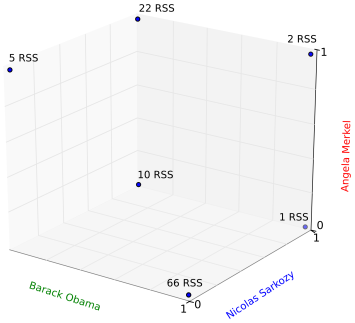
- Different labels but same URI $\rightarrow$ **cross-language feature extraction** : *e.g. Grenada/Grenade* $\rightarrow$ *http ://dbpedia.org/resource/Grenada*

Overview

Feature extraction : Matrix representation and storage

$$\begin{pmatrix} \text{"Obama's approval \ldots} \\ \text{"Protest as Spain \ldots"} \\ \ldots \\ \text{"Libya rebels fight \ldots"} \\ \text{"Obama in Spain \ldots"} \end{pmatrix} \rightarrow \begin{pmatrix} 0 & 0 & \ldots & 1 & 0 \\ 0 & 0 & \ldots & 0 & 1 \\ \ldots & & & & \\ 0 & 1 & \ldots & 0 & 0 \\ 0 & 0 & \ldots & 1 & 1 \end{pmatrix}$$

E.g. The Obama-Merkel-Sarkozy space ...



Results stored in a **relation** (*appears in rss*) in **Cubicweb** :

```
rql('Any X WHERE X appears_in_rss Y')
```

Exploiting information : Clustering news

Creating clusters (groups) of news, using the matrix representation of the data.

Meanshift algorithm (**Scikits-learn**)

- Based on locating the maxima of a density function.
- **Automatically tunes the number of clusters.**

```
from scikits.learn.cluster import MeanShift, estimate_bandwidth
bandwidth = estimate_bandwidth(X, quantile=0.005)
clustering = MeanShift(bandwidth=bandwidth)
clustering.fit(X)
labels = clustering.labels_
```

"Mean shift, mode seeking, and clustering.", Yizong Cheng. IEEE Transactions on Pattern Analysis and Machine Intelligence 1995

$\rightarrow$ Try it !

Other possible alternatives : Ward's algorithm, K-means, . . .

Exploiting information : Queries and Views

Each query returns a **result set (rset)**. A view is called on a **result set** → define the representation rules.

Defining a view (short version)

```
class MyEntitiesView(View):
    __regid__ = 'example-view'

    def call(self):
        rset = self.cw_rset
        for entity in rset.entities():
            do_whatever_you_want_...
```

. . . you just have to plug a rset from a query :

```
rset = rql('Any X WHERE ...')
self.wview('example-view', rset)
```

or within an url :

```
http://myapplication/?rql=Any X WHERE ....&vid=example-view
```

A new approach for querying information from RSS !

All musical artists in the news

```
rql('DISTINCT Any E, R WHERE E appears_in_rss R,
     E has_type T, T label "musical artist"')
```

All living office holder persons in the news

```
rql('DISTINCT Any E WHERE E appears_in_rss R,
     E has_type T, T label "office holder",
     E has_subject C, C label "Living people"')
```

All news that talk about Barack Obama and any scientist

```
rql('DISTINCT Any R WHERE E1 label "Barack Obama",
     E1 appears_in_rss R, E2 appears_in_rss R,
     E2 has_type T, T label "scientist"')
```

All news that talk about a drug

```
rql('Any X, R WHERE X appears_in_rss R,
     X has_type T, T label "drug"')
```

Try it ! . . . with an xml view . . . or a thumbnail view

Vizualisation : Mapping information

View

```
class EntityMapView(View):
    __regid__ = 'map'

    def call(self):
        rset = self.cw_rset
        self.init_map()
        for entity in rset.entities():
            self.add_marker(entity.latitude, entity.longitude,
                            entity.dc_title)
        self.center_and_zoom(0, 0, 1.5)
        self.finish_map()
```

Based on mapstraction (javascript) : http ://mapstraction.com/

✔ **Automatically locate information from RSS news.**

Try it !

Conclusion

Cubicweb and Scikits-learn :

- Efficient and easy-to-use tools for data storing, querying and mining.
- Easy to plug together, only **Python** tools, all **Open Source**.

Using Dbpedia allows to extract very few highly relevant features

- ✓ Decrease the dimensionality of the data.
- ✓ Link features to millions of pages of information.

A new semantic way for querying information

- ✓ **Simple information queries using RQL expressions.**
- ✓ **Use Dbpedia types and categories to refine the selection.**

Future Improvements

Named Entities Recognition

→ use disambiguations links / more refined Regular expressions.

→ add new databases (*MusicBrainz*, *diseasome*, . . .).

→ closely follow Wikipedia with **Dbpedia live update** :

Motorola Mobility (11 :46, 15 August 2011) 'On the 15 August, Google announced that it agreed to acquire the company.'

RSS news analyzing

→ explore new algorithms : bi-clustering, . . .

→ add new data sources : Twitters, Blogs, . . .

→ *'from scikits.learn predict __future__ . . . '* → use **Matrix Completion** to predict new edges in the correlation graph.

Thanks you for your attention !

Questions ?