WikipediaBase

Chris Perivolaropoulos

Sunday 21 February 2016

Contents

1 Introduction

WikipediaBase base is a backend to START responsible for providing access to wikipedia related information. The WikipediaBase we refer to is a python rewrite of the now deprecated Ruby WikipediaBase.

2 People

The python implementation was initially written by Chris Perivolaropoulos and was eventually handed over to

- Alvaro Morales
- Michael Silver

3 Functionality

In WikipediaBase, each (supported) Wikipedia infobox is defined as a class, and each (supported) variable in the infobox is defined as an attribute of that class. All WikipediaBase objects belong by inheritance to the superclass wikibase-term, which supports the attributes `IMAGE-DATA`, `SHORT-ARTICLE`, `URL`, `COORDINATES`, `PROPER`, and `NUMBER`.

WikipediaBase commands and their return values use lisp-like encoding. WikipediaBase provides the following operations:

3.1 get

Given a class, object name, and typed attribute, return the value as a lisp-readable form. Compare Omnibase's get operation.

Valid attribute typecodes are :code (for an attribute name as in infobox wiki markup) and :rendered (for an attribute name in the rendered form of the infobox).

1. Types

   Scripts must return a list of typed values. Valid value typecodes are:

   (a) `:HTML`

   A string suitable for rendering as paragraph-level HTML. The string must be escaped for lisp, meaning double quoted, and with double quotes and backslashes escaped with backslashes. The string is not required to contain any HTML codes. For example:

```
(get "wikipedia-sea" "Black Sea" (:code "AREA
   "))
=> ((:html "436,402 km2 (168,500 sq mi)"))

(get "wikipedia-president" "Bill Clinton" (:
   code "SUCCESSOR"))
=> ((:html "George W. Bush"))

(get "wikipedia-president" "Bill Clinton" (:
   rendered "Succeeded by"))
```

```
=> ((:html "George W. Bush"))
```

(b) `:YYYYMMDD`

Parsed dates are represented as numbers, using YYYYMMDD format with negative numbers representing B.C. dates. (Unparsable dates are represented as HTML strings using the :HTML typecode.)

```
(get "wikibase-person" "Barack Obama" (:ID "
   BIRTH-DATE"))
=> ((:yyyymmdd 19610804))

(get "wikibase-person" "Julius Caesar" (:ID "
   BIRTH-DATE"))
=> ((:YYYYMMDD -1000713))
```

(c) `:CALCULATED`

Typecode for attributes calculated by WikiBase based on characteristics of the article, e.g., *GENDER* and *NUMBER*. See below under Special Attributes for a complete list of calculated attributes.

(d) `:CODE` Deprecated old synonym for `:HTML`.

(e) `:STRING` Deprecated old synonym for `:HTML`.

2. Special Attributes

Some attributes are special because they are computed by WikipediaBase rather than being fetched from infoboxes, or rather than being fetched directly. These attributes should be specific to wikibase-term, wikibase-person, and wikipedia-paragraphs.

(a) `SHORT-ARTICLE`, `wikibase-term`

The first paragraph of the article, or if the first paragraph is shorter than 350 characters, then returns the first paragraphs such that the sum of the rendered characters is at least 350.

(b) `URL`, `wikibase-term`

Returns the URL of the article as `((:url URL))`

(c) `IMAGE-DATA`, `wikibase-term`

Returns a list of URLs for images in the article content (excludes images that are in the page but outside of the article content). If there are no images, should return an empty list. The "best" image should be the first URL in the list; if there is a picture at the top of the infobox, this is considered to be the best image, or otherwise the first image that appears anywhere in the article. If there is no caption, the caption value should be omitted, e.g., `((0 "Harimau_Harimau_cover.jpg"))` rather than `((0 "Harimau_Harimau_cover.jpg" ""))`.

(d) `COORDINATES`, `wikibase-term`

Computed from latitude and longitude attributes given in the article or, if none can be found, the infobox. The value is a list of the latitude and longitude, e.g., ((:coordinates latitude longitude))

(e) `BIRTH-DATE`, `wikibase-person`

Fetched from the infobox, or, if it is not found, from the artle, or, if it is not found, the category information of the article. Always relies on the first date of birth found, matching one of several supported formats. If this attribute has a value, then the object is considered to be a person with respect to the GENDER attribute (see below). The value can be a parsed or unparsed date. Parsed dates are represented as numbers, using YYYYMMDD format with negative numbers representing B.C. dates. Unparsed dates are strings.

(f) `DEATH-DATE`, `wikibase-person`

Fetched similarly to BIRTH-DATE. Returns the same value types as BIRTH-DATE, except if the person is still alive, throws an error with the reply "Currently alive".

(g) `GENDER`, `wikibase-person`

Computed from the page content based on heuristics such as the number of times that masculine vs. feminine pronouns appear. Valid values are `:masculine` and `:feminine`.

(h) `NUMBER`, `wikibase-term`

Computed from the page content based on heuristics such as number of times the page's title appears plural. Valid for all objects. Returns `#t` if many, `#f` if one.

i. `PROPER`, `wikibase-term`

Computed from the page content based on heuristics such as number of times the page's title appears capitalized when not

at the start of a sentence. Valid for all objects. Returns `#t` if proper and `#f` if not.

3.2 get-classes

Given an object name, return a list of all classes to which the object belongs, with classes represented as lisp-readable strings. Class names are conventionally given in lower case, but this is not an absolute requirement. E.g.,

```
(get-classes "Cardinal (bird)")
=> ("wikibase-term" "wikipedia-paragraphs" "wikipedia
   -taxobox")

(get-classes "Hillary Rodham Clinton")
=> ("wikibase-term" "wikipedia-paragraphs" "wikibase-
   person" "wikipedia-officeholder" "wikipedia-person
   ")
```

3.3 get-attributes

Given a class name, return a list of all attributes that the class implements (that is, all variables that the infobox implements), as lisp-readable strings. Also sometimes given is the human-readable rendering of the attribute and/or the value typecode for the attribute. Attribute names are conventionally given in upper case, but this is not an absolute requirement. E.g.,

```
(get-attributes "wikipedia-officeholder" "Barack
   Obama")
=> ((:CODE "TERM_END3" :VALUE :YYYYMMDD) ...)
```

3.4 ort-symbols

`sort-symbols` takes any number of symbols and sorts them into subsets by the length of the associated article. E.g.,

```
(sort-symbols  "Obama (surname)" "Barack Obama")
=> (("Barack Obama") ("Obama (surname)"))
```

3.5 `sort-symbols-named`

`sort-symbols-named` takes a synonym and any number of symbols and sorts the symbols into subsets; if any symbol name is the same as the synonym, it and its subset are sorted to the front. (This should be a case insensitive match, but is it? And again, what's with the subsets?) E.g.

```
(sort-symbols-named "cake" "Cake (TV series)" "Cake (
    firework)" "Cake (film)" "Cake (drug)"
"Cake" "Cake (band)" "Cake (advertisement)" "The Cake
    ")
=> (("Cake") ("Cake (band)") ("Cake (advertisement)")
     ("Cake (TV series)")
("The Cake") ("Cake (film)") ("Cake (firework)") ("
    Cake (drug)"))
```

4 Getting started

The WikipediaBase implementation that we refer to is written in python. Previous implementations were written in Java and Ruby but the language of choice for the rewrite was python for multiple reasons:

- Python is in the pre-graduate curriculum of MIT computer science department. This will ease the learning curve of new members of Infolab.
- Python is a easy to learn and mature language with a rich and evolving ecosystem. This fact eases the introduction of new maintainers even further.

The entire WikipediaBase resides in a git repository in infolab's github orginization page

```
git clone git@github.com:infolab-csail/WikipediaBase
```

WikipediaBase depends on multiple other python packages. Fortunately, python is shipped not only with a great package manager, but also with a mechanism called virtualenv that isolates installations of a project's dependencies from the rest of the system, thus avoiding problems like version or namespace collisions. The way this effectively works is that the global python installation is half copied half symlinked to a local directory and the

dependencies are installed only in the local sandbox. To create and activate a python virtualenv:

```
$ virtualenv --no-site-packages py
$ . py/bin/activate
$ which python
/the/local/directory/py/bin/python
```

Now that we can safely install anything we want without breaking any global installation

```
pip install -r requirements.txt
```

We will need some extra stuff for WikipediaBase to work:

- Postresql
- Redis

The installation process of these packages varies across platforms. Both are databases. Their purpose is for caching repeated computations and for storing ahead-of-time computation like infobox markup name to rendered name maps and synonyms.

5 Architecture

5.1 Infobox

Infoboxes are tables that are commonly used in wikipedia to provide an overview of the information in an article in a semi structured way. Infoboxes are the main source of information for WikipediaBase.

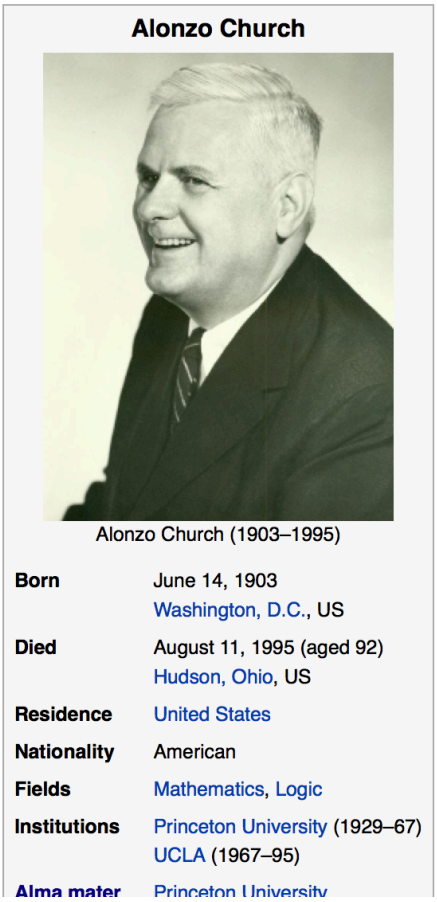

Figure 1: An example of an infobox

In mediawiki markup terms an infobox is a markup template with a type that gets rendered into html so that the provided information makes sense in the context that it is provided. For example:

```
{{Infobox scientist
| name              = Gerhard Gentzen
| image             = Gerhard Gentzen.jpg
| image_size        =
| alt               =
| caption           = Gerhard Gentzen in Prague,
   1945.
| birth_date        = {{Birth date|1909|11|24}}
| birth_place       = [[Greifswald]], [[Germany]]
```

```
| death_date         = {{Death date and age
   |1945|8|4|1909|11|24}}
| death_place        = [[Prague]], [[Czechoslovakia]]
| nationality        = [[Germany|German]]
| fields             = [[Mathematics]]
| workplaces         =
| alma_mater         = [[University of Gottingen]]
| doctoral_advisor   = [[Paul Bernays]]
| doctoral_students  =
| known_for          =
| awards             =
}}
```

will yield:

Figure 2: An example of an infobox

Infobox types are organized into a fairly wide hierarchy. For example Template:Infobox Austrian district is a special case of a Template:Infobox settlement and each is rendered differently. For our purposes, and to mirror the markup definition of infoboxes, an infobox I with attributes a_i and values v_i is a set of pairs (a_i, v_i) together with a infobox type t. Each attribute a_i and value v_i have two forms:

- a rendered form, a_i^r and v_i^r respectively, which is the rendered HTML representation and
- a markup form, a_i^m and v_i^m which is the mediawiki markup code that corresponds to them.

An article may have more than one infoboxes, for example Bill Clinton article has both Infobox Officeholder and Infobox President infoboxes.

The `Infobox` class is the basic data type for accessing information from the infobox of an article. `Infobox`, as well as `Article`, are what one would use were they to use wikipediabase as a python library. The methods provided by an infobox are:

types Because we retrieve an infobox based on a symbol name (ie page name), a single `Infobox` may actually be an interface for multiple infoboxes. There is a separate method, based on this one, for getting types in a format suitable for START.

Value access is possible provided either a_i^r or a_i^m.

Rendered keys are provided using the `MetaInfobox` (see below).

Infobox export to python types, namely:

- dict for $a_i^r \rightarrow v_i^r$ or $a_i^m \rightarrow v_i^m$
- the entire infobox rendered, or in markup form.

5.2 Infobox tree

5.3 MetaInfobox

The `MetaInfobox` is a subclass of the `Infobox` that provodes information about the infobox, most importantly a map between markup attributes. Say we have an infobox of type I which has attributes $a_1, ..., a_n$. Each instance of that infobox I defines

It is an infobox with all the valid attributes and each value is all the names of all attributes that are equivalent to them. Eg An infobox of type Foo that has valid attributes A, B, C and D and A, B and C are equivalent has a meta infobox that looks something like:

Attribute	Value
A	!!!A!!! !!!B!!! !!!C!!!
B	!!!A!!! !!!B!!! !!!C!!!
C	!!!A!!! !!!B!!! !!!C!!!
D	!!!D!!!

5.4 Article

The `Article` data structure is responsible for accessing any resource relevant to the article at large.

5.5 Fetcher

The fetcher is an abstraction over the communicatioln of WikipediaBase with the outside world. It is a singleton object that implements a specific interface.

Fetchers are organized in an inheriting hierarchiy

BaseFetcher The baseclass for fetchers, it will return the symbol instead of trying to resolve it in any way

Fetcher contains the core functionality of a a fetcher. It will fetch articles from *wikipedia.org*. It is possible to direct it to a mirror but wikipedia-mirror's runtime performance turned out to be prohibitive.

CachingFetcher inherits fetcher and retains it's functionality, only it uses Redis to cache the fetched symbols. It is the default fetcher for wikipediabase.

StaticFetcher is a class that implements the `BaseFetcher` interface but instead of reaching out to some data source for the data the return values are statically defined. It is used most notably by `MetaInfobox` to use the `Infobox` functionality to convey arbitrary information.

By default, markup is fetched from the backend. If $\text{force}_{\text{live}}$ is set to True, the markup will be fetched from live wikipedia.org

When tests are ran on TravisCI, we always want to use live data. We check if Travis is running tests by looking at the $\text{WIKIPEDIABASE}_{\text{FORCELIVE}}$ env variable.

5.6 Renderer

Renderers are singleton classes that are useful for rendering mediawiki markup into HTML. Originally the wikiepedia sandbox was used by wikipediabase for rendering pages because it is slightly faster than the API, but the wikipedia-mirror was really slow at this and wikipedia.org would consider it an abuse of the service and block our IP. For that reason we eventually switched to the API with Redis caching, which works out pretty well because `Renderer` objects end up being used only my `MetaInfobox` which has quite a limited scope, making thus cache misses rarely.

5.7 Caching

TODO, check what alvaro did with this

5.8 Logging

5.9 Utilities

1. General
2. Database utilities

5.10 Pipeline

When resolving a query WikipediaBase employs a pipeline of modules to figure out what the best way to respond would be.

1. Frontend

   WikipediaBase can be used as a library but it's primary function is as a backend to START. The communication between START and WikipediaBase is carried out over a plaintext telnet connection on port {port} using EDN-like sexpressions. The frontend handles the network connection with START, translates the received queries into calls to knowledgebase and then translate the knowledgebase response into properly formulated sexpressions that it sends back over the telnet connection.

2. Knowledgebase

   The knowledgebase is the entry point to the rest of wikipediabase. It uses the Provider/Acquirer pattern to transaprently provide the frontend with arbitrary methods. Those methods are responsible for chosing whether we are to resort to classifiers or resolvers (or any other mechanism) for answering the query. Available classifiers and resolvers become accessible to the knowledgebase automatically using their base class.

3. Classifiers

   Each classifier is a singleton that implements a heuristic for deducing a set of classes of an object. An object may inhibit zero or more classes. There are a couple classifiers available at the moment. Typically a classifier will only deduce whether an object actually inhibits a specific class or not but that is not necessary.

   (a) Term
   The `TermClassifier` simply assigns the `wikipedia-term` class. Wikipediabase only deals with wikipedia related information.

(b) Infobox

The `InfoboxClassifier` assigns to a term the classes of the infobox. For example Bill Clinton's page contains the infobox:

```
    {{Infobox president
|name          = Bill Clinton
|image         = 44 Bill Clinton 3x4.jpg{{!}}border
|office        = [[List of Presidents of the United States|42nd]] [[President
|vicepresident = [[Al Gore]]
|term_start    = January 20, 1993
|term_end      = January 20, 2001
|predecessor   = [[George H. W. Bush]]
|successor     = [[George W. Bush]]
|order1        = 40th and 42nd [[List of Governors of Arkansas|Governor of Ar
|lieutenant1   = [[Winston Bryant]]<br>[[Jim Guy Tucker]]
|term_start1   = January 11, 1983
|term_end1     = December 12, 1992
|predecessor1  = [[Frank D. White]]
|successor1    = [[Jim Guy Tucker]]
|lieutenant2   = [[Joe Purcell]]
|term_start2   = January 9, 1979
|term_end2     = January 19, 1981
|predecessor2  = [[Joe Purcell]] {{small|(Acting)}}
|successor2    = [[Frank D. White]]
|office3       = 50th [[Arkansas Attorney General|Attorney General of Arkansas
|governor3     = [[David Pryor]]<br>[[Joe Purcell]] {{small|(Acting)}}
|term_start3   = January 3, 1977
|term_end3     = January 9, 1979
|predecessor3  = [[Jim Guy Tucker]]
|successor3    = Steve Clark
|birth_name    = William Jefferson Blythe III
|birth_date    = {{birth date and age |1946|8|19}}
|birth_place   = [[Hope, Arkansas|Hope]], [[Arkansas]], [[United States|U.S.]]
|death_date    =
|death_place   =
|party         = [[Democratic Party (United States)|Democratic]]
|spouse        = {{marriage|[[Hillary Clinton|Hillary Rodham]]|October 11, 19
|relations     = ''See [[Clinton family]]''
|children      = [[Chelsea Clinton|Chelsea]]
|parents       = [[William Jefferson Blythe, Jr.]]<br>[[Virginia Clinton Kell
```

```
|alma_mater    = [[Edmund A. Walsh School of Foreign Service|Georgetown Unive
|religion      = [[Baptists|Baptist]] {{small|(formerly [[Southern Baptist Co
|signature     = Signature of Bill Clinton.svg
|signature_alt = Cursive signature of Bill Clinton in ink
|website       = {{url|clintonlibrary.gov|Library website}}
}}
```

And therefore gets the class `wikipedia-president`.

(c) Person

`PersonClassifier` assigns the class `wikibase-person` using a few heretics in the order they are described:

i. Category regexes
Use the following regular expressions to match categories of an article.

- `.* person`
- `^\d+ deaths.*`
- `^\d+ births.*`
- `.* actors`
- `.* deities`
- `.* gods`
- `.* goddesses`
- `.* musicians`
- `.* players`
- `.* singers`

ii. Category regex excludes
Exclude the following regexes.

- `\sbased on\s`
- `\sabout\s`
- `lists of\s`
- `animal\`

iii. Category matches
We know an article refers to a person if the page is in one or more of the following mediawiki categories:

- `american actors`
- `american television actor stubs`
- `american television actors`

- `architects`
- `british mps`
- `character actors`
- `computer scientist`
- `dead people rumoured to be living`
- `deities`
- `disappeared people`
- `fictional characters`
- `film actors`
- `living people`
- `musician stubs`
- `singer stubs`
- `star stubs`
- `united kingdom writer stubs`
- `united states singer stubs`
- `writer stubs`
- `year of birth missing`
- `year of death missing`

For example Leonardo DiCaprio's page has the following categories:

- Leonardo DiCaprio
- 1974 births
- **Living people**
- 20th-century American male actors
- 21st-century American male actors
- American environmentalists
- American film producers
- American male child actors
- American male film actors
- American male soap opera actors
- American male television actors
- American people of German descent
- American people of Italian descent
- American people of Russian descent
- American philanthropists

- Best Actor AACTA Award winners
- Best Actor Academy Award winners
- Best Drama Actor Golden Globe (film) winners
- Best Musical or Comedy Actor Golden Globe (film) winners
- California Democrats
- Film producers from California
- Formula E team owners
- Male actors from Hollywood, California
- Male actors from Palm Springs, California
- Male actors of Italian descent
- People from Echo Park, Los Angeles
- Silver Bear for Best Actor winners

As it is obvious the list of categories is arbitrary and very far from complete. Multiple methods have been considered for fixing this. Some of them are:

- Supervised machine learning methods like SVM using other methods of determining person-ness to create training sets.
- Hand-pick common categories for person articles determined again with the other criteria

4. Resolvers

   Resolvers are also singletons but their purpose is to find the value of the requested property.

   (a) Error

   (b) Infobox

   (c) Person

   (d) Section

   (e) Term

   If the article matches one or more of the following categories:

5. Lisp types

   Lisp type instances are wrappers for python objects or values that are presentable in s-expression form that START can understand. They

are created either from the raw received query and unwrapped to be useful to the pipeline, or by the answer WikipediaBase comes up with and then encoded into a string sent over telnet to START.

6 Provider/Acquirer model

WikipediaBase attempts to be modular and extendible. To accomplish this it is often useful to multiplex multiple sources of the same type of data resource. This is particularly useful when accessing heuristic methods like classifier. To promote modularity and to avoid hard dependencies the provider/acquirer model was created:

A `Provider` is an object though which we can access resources that are stored in a key-value fashion. The `Provider` class offers facilities like decorators to make this provision easy. An `Acquirer` has transparent access to the resources of multiple `Provider` s as if they were a single key value store. This pattern is most notably used for the `KnowledgeBase` to provide the `Frontend` with the way of accessing resoruces.

6.1 TODO Example

7 Testing

7.1 Unit testing

The good functioning of WikipediaBase is assured by a comprehensive test suite of unit tests, functional tests and regression tests.

1. Unit tests

   Unit tests test small blocks of functionality, that are composed to create the system at large. For unit testing we use python's default testing library. Each test is a class the subclasses

2. Functional and regression tests

   Functional tests are tests written before, during or shortly after the development of a system and they assert the correct overall functioning of the system. Regression tests are very akin to functional tests. They prove that a found bug was fixed and assert that it will not appear again later. Functional and regression tests currently reside in `tests/examples.py`

7.2 TODO Examples

8 Synonyms

8.1 Good/Bad synonyms

8.2 Synonym generation

9 Backend databases

9.1 DBM

9.2 SQLite

9.3 Redis

9.4 Postgres

10 Data sources

11 Date parser

Dateparser resides in a separate package called overlay-parse

11.1 Parsing with overlays

The concept of an overlay was inspired by emacs overlays. They are objects that specify the behavior of a subset of a text, by assigning properties to it. An overlay over a text t in our context is tuple of the range within that text, a set of tags that define semantic sets that the said substring is a member of, and arbitrary information (of type A) that the underlying text describes. More formally:

$$o_i \in TextRanget \times Set(Tag) \times A \qquad numbers$$
$$Text \to \{o_1, o_2, ..., o_n\}$$

So for example out of the text

$$The\,weather\,today,\ \overbrace{Tuesday}^{o_1}\ \overbrace{21^{st}}^{o_2}\ of\ \overbrace{November}^{o_3}\ \overbrace{2016}^{o_4},\ was\,sunny.$$

We can extract overlays $\{o_1, ..., o_4\}$, so that

$$
\begin{array}{lll}
o_1 = (& r(\text{"}Tuesday\text{"}), & \{\text{DayOfWeek}, \text{FullName}\}, \quad 2) \\
o_2 = (& r(\text{"}21^{st}\text{"}), & \{\text{DayOfMonth}, \text{Numeric}\}, \quad 21) \\
o_3 = (& r(\text{"}November\text{"}), & \{\text{Month}, \text{FullName}\}, \quad 11) \\
o_4 = (& r(\text{"2016"}), & \{\text{Year}, \text{4digit}\}, \quad 2016)
\end{array}
$$

Notice how for all overlays of the example we have $A = \mathbb{N}$, as we encode day of the week, day of the month, month and year as natural numbers. We encode more precise type information (ie that a day is inherently different than a month) in the tag set.

Once we have a set of overlays we can define overlay sequences as overlays whose ranges are consecutive, that is their and their tag sets match particular patterns. For example we can search for sequences of overlays that match the pattern

$$p = \text{DayOfMonth}, \text{Separator}(/), (\text{Month} \wedge \text{Number}), \text{Separator}(/), \text{Year}$$

to match patterns like 22/07/1991, where $Separator(/)$ matches only the character "/"

11.2 The implementation

1. Comparison

11.3 The dates example

11.4 Benchmarks

12 Future

12.1 Configuration

1. Persistence
2. Pass by reference
3. Lenses
4. Laziness
   (a) Referential (Ref - Items)
   (b) Computational