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" "
```

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 (dr
"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

4.1 Language

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.

4.2 Getting the code

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

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

4.3 Virtualenv and dependencies

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 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.

   (a) Protocol

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 assigning a class of an object. Thereare a couple classifiers available at the moment.

4. Resolvers

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

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.

5.2 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.

5.3 Infobox

5.4 Article

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:

$$\begin{aligned}&o_i \in TextRanget \times Set(Tag) \times A\\&Text \rightarrow \{o_1, o_2, ..., o_n\}\end{aligned}$$

So for example out of the text

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

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

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

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

11.3 Optimization

1. Comparison

11.4 The dates example

11.5 Benchmarks

12 Future

12.1 Configuration

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

12.2 START deployment

12.3 Test suites

12.4 Bugs

12.5 Answer hierarchy