Extracting relational data from wikipedia

Chris Perivolaropoulos

Sunday 21 February 2016

Contents

1 Introduction

1.1 START

The START Natural Language System is a software system designed to answer questions that are posed to it in natural language. START parses incoming questions, matches the queries created from the parse trees against its knowledge base and presents the appropriate information segments to the user. In this way, START provides untrained users with speedy access to knowledge that in many cases would take an expert some time to find.

START (SynTactic Analysis using Reversible Transformations) was developed by Boris Katz at MIT's Artificial Intelligence Laboratory. Currently, the system is undergoing further development by the InfoLab Group, led by Boris Katz. START was first connected to the World Wide Web in December, 1993, and in its several forms has to date answered millions of questions from users around the world.

A key technique called "natural language annotation" helps START connect information seekers to information sources. This technique employs natural language sentences and phrases annotations as descriptions of content that are associated with information segments at various granularities. An information segment is retrieved when its annotation matches an input question. This method allows START to handle all variety of media, including text, diagrams, images, video and audio clips, data sets, Web pages, and others.

The natural language processing component of START consists of two modules that share the same grammar. The understanding module analyzes English text and produces a knowledge base that encodes information found in the text. Given an appropriate segment of the knowledge base, the generating module produces English sentences. Used in conjunction with the technique of natural language annotation, these modules put the power of sentence-level natural language processing to use in the service of multimedia information access.

2 Wikipediabase

2.1 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.

2.2 People

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

- Alvaro Morales
- Michael Silver

2.3 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:

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).

   (a) Types

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

   i. `: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"))
```

ii. `: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))
```

iii. `: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.

iv. `:CODE` Deprecated old synonym for `:HTML`.

v. `:STRING` Deprecated old synonym for `:HTML`.

(b) 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.

i. `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.

ii. `URL`, `wikibase-term`

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

iii. `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" ""))`.

iv. `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))

v. `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.

vi. `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".

vii. `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`.

viii. `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.

    A. `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.

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-officehol
```

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) ...)
   ```

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)"))
   ```

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)" "Ca
   "Cake" "Cake (band)" "Cake (advertisement)" "The Cake")
   => (("Cake") ("Cake (band)") ("Cake (advertisement)") ("Cake (TV series)")
   ("The Cake") ("Cake (film)") ("Cake (firework)") ("Cake (drug)"))
   ```

2.4 Getting started

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.

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
   ```

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.

2.5 Architecture

1. Pipeline

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

   (a) 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.

      i. Protocol

   (b) 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.

   (c) 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.

   (d) Resolvers

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

   (e) 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.

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.

3. Infobox

4. Article

2.6 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.

1. **TODO** Example

2.7 Testing

1. Unit testing

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

   (a) 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

   (b) 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`

2. **TODO** Examples

2.8 Synonyms

1. Good/Bad synonyms

2. Synonym generation

2.9 Backend databases

1. DBM

2. SQLite

3. Redis

4. Postgres

2.10 Data sources

2.11 Date parser

Dateparser resides in a separate package called overlay-parse

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:

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

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

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

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 "/"

2. The implementation

   (a) Comparison

3. The dates example

4. Benchmarks

2.12 Future

1. Configuration
   (a) Persistence
   (b) Pass by reference
   (c) Lenses
   (d) Laziness
      i. Referential (Ref - Items)
      ii. Computational
2. START deployment
3. Test suites
4. Bugs
5. Answer hierarchy

3 WikipediaMirror

wikipedia-mirror is a tool for generating mirrors of wikipedia.org using the dumps provided by wikipedia.org.

3.1 Mediawiki stack overview

Wikipedia-mirror builds upon the mediawiki stack provided by bitnami. A service that builds the entire server within the confines of a directory. This is useful because we avoided the overhead of dealing with container or VM technologies and we had direct access to the filesystem of the stack while still having bitnami's build system do the tedious job of orchestrating the various components and separating our sever from the rest of the system.

The stack is comprised of

- An http server, in our case apache
- The web application runtime, in our case PHP
- A database, in our cas MySQL
- The web application itself, in our case mediawiki

All of the above are provided by the the bitnami mediawiki stack. Xampp used to be go-to for that but it is unmaintained so we decided to go with bitnami which works pretty well.

Once the stack is set up properly the wikipedia dump xml is downloaded and then turned into an sql dump with mwdumper. Could be piped directly to MySQL? but extracting can take time and things tend to go wrong during the dumping step.

1. Elements of the stack We present each of the elements of the stack in more detail below.

   (a) Apache
   As per wikipedia:

   ```
   The Apache HTTP Server, colloquially called Apache, is the world's
   most used web server software. Originally based on the NCSA HTTPd
   server, development of Apache began in early 1995 after work on the
   NCSA code stalled. Apache played a key role in the initial growth of
   the World Wide Web, quickly overtaking NCSA HTTPd as the dominant HTTP
   server, and has remained most popular since April 1996. In 2009, it
   became the first web server software to serve more than 100 million
   websites.

   Apache is developed and maintained by an open community of developers
   under the auspices of the Apache Software Foundation. Most commonly
   used on a Unix-like system (usually Linux), the software is available
   for a wide variety of operating systems besides Unix, including
   eComStation, Microsoft Windows, NetWare, OpenVMS, OS/2, and
   TPF. Released under the Apache License, Apache is free and open-source
   software.
   ```

   it is fair to say that apache is at least one of the most popular web servers on the internet. wikipedia.org itself seems to be using a more complex stack involving varnish, an HTTP accelerator, and nginx, an alternative, also quite popular HTTP server. We arrive at this conclusion by inspecting the headers returned by wikipedia.org. In the `http://www.wikipedia.org` case we are redirected to the secure domain (pay attention to the `Server:` line):

   ```
   $ curl -s -D - http://www.wikipedia.org -o /dev/null
   ```

```
HTTP/1.1 301 TLS Redirect
Server: Varnish
[...]
```

And if we directly ask for `https://www.wikipedia.org` nginx seems to be handling our request:

```
$ curl -s -D - https://www.wikipedia.org -o /dev/null
HTTP/1.1 200 OK
Server: nginx/1.9.4
[...]
```

However it is beyond the scope of the project to precisely replicate wikipedia's infrastructure. We focus on the functionality. Therefore due to the popularity, familiarity and by virtue of apace being part of the automatically installable bitnami mediawiki stack, we use it as our server.

(b) PHP

Mediawiki, which is discussed later, is written entirely in PHP, a popular server side, dynamically typed, object oriented scripting language. PHP is essential and is installed along the bitnami mediawiki stack. PHP is popular among web developers partly due to it's support for multiple relational database libraries (including PostgreSQL, MySQL, Microsoft SQL Server and SQLite) and it essentially being structred as a template language generating HTML.

(c) MySQL

Mediawiki can use a number of different SQL database backends:

- **MSSQL:** An SQL database by Microsoft
- **MySQL:** Using the standard PHP library for MySQL.
- **MySQLi:** An extension to the MySQL backend
- **Oracle:** A propertiary SQL database by Oracle.
- **SQLite:** An SQL database that is typically accessed as a library rather than over a client-server scheme as is the case with the other options on the list.

Wikipedia provides multiple dump files for SQL tables of secondary importance in MySQL format (eg. page redirects, categories etc) and suggests `mwdumper` which parses the XML dumpls of the wikipedia articles into MySQL. That and bitnami providing it as part of it's automatically built stack, make MySQL the obvious choice for the wikipedia-mirror stack.

   (d) MediaWiki

   Mediawiki is the beating heart of wikipedia.

2. Tools

   A number of tools were developed in assisting the

   (a) page$_{\text{remover.c}}$
   (b) sql-clear.sh
   (c) utf8thread.c
   (d) webmonitor.py
   (e) xml-parse.sh

3. Setting up

   Following are step by step instructions First, clone the git repo:

```
$ git clone https://github.com/fakedrake/wikipedia-mirror
$ cd wikipedia-mirror
```

   At this point in theory one can run `make sql-load-dumps` which will take care of stting up everything needed to load the the database dumps into the working SQL database. Of course for that to happen first a couple of steps need to be carried out:

   - Download the wikipedia database dumps in XML format.
   - Transform them into a format that MySQL understands.
   - Set up the bitnami stack that includes a local install of MySQL
   - Load the MySQL dumps into MySQL

   All of these steps are encoded as part of the a dependency hierarchy encoded into makefile targets and are in theory taken care of automatically, effectively yielding a functioning wikipedia mirror. However this process is extremely long fragile so it is advised that each of these steps be run individually by hand.

   First, download and install bitnami. The following command will fetch an executable from the bitnami website and make a local installation of the bitnami stack discussed above:

```
$ make bmw-install
```

Next step is to make sure `maven`, the java is a software project management and comprehension is installed, required to install and setup mwdumper (see below). You can do that by making sure the following succeeds:

```
$ mvn --version
```

Note: if running on Ubuntu 14.04, you may need to install Maven (for Java) using `sudo apt-get install maven`.

Now everything is installed to automatically download Wikipedia's XML dumps and then convert them to SQL using maven. First maven will be downloaded and built. Then the compressed XML dumps will be downloaded from the wikipedia, they will be uncompressed and finally converted to MySQL dumps using `mwdumper`. This is a fairly lengthy process taking 6 to 11 hours on a typical machine:

```
$ make sql-dump-parts
```

After that's done successfully you can load the SQL dumps to the MySQL database.

```
$ make sql-load-parts
```

Finally the

```
$ make mw-extensions
```

3.2 Mediawiki Extensions

For mediawiki to act like wikipedia a number of extensions are required. The installation process of such extensions is not automated or streamline. To automatically manage this complexity a mechanism is provided for declaratively installing extensions. To add support for an extension to wikipediabase one needs to add the following code in `Makefile.mwextnesions` (modifying accordingly):

```
MW_EXTENSIONS += newextension
mw-newextension-url = url/to/new/extnesion/package.tar.gz
mw-newextension-php = NewExtensionFile.php
mw-newextension-config = '$$phpConfigVariable = "value";'
```

And wikipedia-mirror will take care of checking if the extension is already installed and if not it will put the right files in the right place and edit the appropriate configuration files. The entry points for managing extensions are (provided that the name of the registered extension is newextension):

```
make mw-print-registered-extensions # Output a list of the registed extensions
make mw-newextension-enable         # Install and/or enable the extension
make mw-newextension-reinstall      # Reinstall an extension
make mw-newextension-disable        # Disable the extension
make mw-newextension-clean          # Remove the extension
```

All registered extensions will be installed and enabled when wikipedia-mirror is built.

3.3 Dumps

Wikipedia provides monthly dumps of all it's databases. The bulk of the dumps come in XML format and they need to be encoded into MySQL to be loaded into the wikipedia-mirror database. There are more than one ways to do that.

1. PHP script

   Mediawiki ships with a utility for importing the XML dumps. However it's use for importing a full blown wikipedia mirror is discouraged due to performance tradeoffs. Instead other tools like mwdumper are recommended that transform the XML dump into MySQL queries that populate the database.

2. mwdumper

   The recomended tool for translating the XML dumps into MySQL code is mwdumper. Mwdumper is written in java and is shipped separately from mediawiki.

   (a) Xml sanitizer
   (b) Article dropper

3.4 Automation

1. Makefiles / laziness
2. Shell scripts
3. Bitnami

3.5 Performance

1. Compile time

   Compile time includes the time it takes for:

   - Downloading all the components of a wikipedia server
   - The bitnami stack
     - mwdumper
     - mediawiki-extensions
     - Installing and building those components (~1 min)
     - Downloading the wikipedia dumps
     - Preprocessing the dumps (~10 mins)
     - Populating the mysql database (~10 days)
       Builds were done on Infolab's Ashmore. The system's specs are quite high end but the bottleneck was the disk IO so less than 1% of the rest of the available resources were used during the MySQL database population.

   (a) Attempts to optimizing MySQL

2. Runtime

   Runtime of wikipedia mirror turned out to be too slow to be useful and therefore the project was eventually abandoned. Namely for the full wikipedia dump of July 2014 the load time for the Barack Obama, not taking advantage of caching was at the order of ~30s.

4 Related CSAIL projects

5 This document

This document was written using emacs org-mode.