Extracting relational data from Wikipedia

Chris Perivolaropoulos

Sunday 21 February 2016

Contents

1 Acknowledgements

2 Abstract

MiT InfoLab's START (SynTactic Analysis using Reversible Transformations) is the worlds first question answering system. It was developed in the early 80s and went online in 1993. For accessing most data sources it takes advantage of, START depends on Omnibase, the *"virtual database"* providing uniform access to multiple sources on the web. Wikipedia surfaced at about the same time as Omnibase, during the early 00s, and did not gain popularity until the mid 00s. So, while wikipedia can provide a vast amount of information, Omnibase does not include it in it's data sources. Our work is divided in two parts: WikipediaBase, and wikipedia mirror. WikipediaBase provides START access to information in wikipedia with an interface similar to Omnibase's. Wikipedia mirror is a program that automates the creation of wikipedia.org mirrors on a local machine, to provide control and unrestricted access to the dataset without depending or abusing wikipedia.org.

3 Introduction

3.1 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.

3.2 Omnibase

3.3 Wikipedia

3.4 WikipediaBase

3.5 Wikipedia mirror

4 Wikipediabase

4.1 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.

4.2 People

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

- Alvaro Morales
- Michael Silver

4.3 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-officeholder" "
   wikipedia-person")
```

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)" "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)"))
```

4.4 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.

4.5 Architecture

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

2. Infobox tree

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!!!

4. Article

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

5. Fetcher

   The fetcher is an abstraction over the communicatioln of Wikipedia-Base 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.

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.

7. Caching

   TODO, check what alvaro did with this

8. Logging

9. Utilities

   (a) General

   (b) Database utilities

10. 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.

    (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 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.

    i. Term

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

ii. 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]] [[Presid
|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
|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 Arka
|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.
|death_date    =
|death_place   =
|party         = [[Democratic Party (United States)|Democratic]]
|spouse        = {{marriage|[[Hillary Clinton|Hillary Rodham]]|October 11,
|relations     = ''See [[Clinton family]]''
|children      = [[Chelsea Clinton|Chelsea]]
|parents       = [[William Jefferson Blythe, Jr.]]<br>[[Virginia Clinton K
```

```
|alma_mater    = [[Edmund A. Walsh School of Foreign Service|Georgetown Un
|religion      = [[Baptists|Baptist]] {{small|(formerly [[Southern Baptist
|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`.

iii. Person

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

A. 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`

B. Category regex excludes

Exclude the following regexes.

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

C. 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

(d) Resolvers

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

i. Error
ii. Infobox
iii. Person
iv. Section
v. Term

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

(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.

4.6 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

4.7 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

4.8 Synonyms

Before we talk about synonyms it is important to concretely define symbols in the context of the omnibase universe:

> Symbols are identifiers of "objects" in a data source. (The term "symbol" is unfortunate, since it has so many meanings in computer science, but we're stuck with it for historical reasons.)

Since language tends to have multiple ways of referring to the same things, defining aliases for symbols is imperative.

> Synonyms are names which users can use to refer to symbols. (The term "synonym" is unfortunate, because this is really a one-way mapping -"gloss" would be a better term but we're stuck with "synonym" for hysterical raisins.)

The definition of synonyms is the job of the backend itself. Therefore it is the job of WikipediaBase to define the set of synonyms required.

1. Good/Bad synonyms

   There are rules to what is considered a good and what a bad synonym. In short synonyms:

   - Should not lead with articles ("the", "a", "an")
   - Should not lead with "File:" or "TimedText:".
   - Should not fragment anchors. Eg "$\text{Alexander}_{\text{Pushkin}}\#\text{Legacy}$"
   - Should not start with the following:
     - "List of "
     - "Lists of "
     - "Wikipedia: "
     - "Category: "
     - ":Category: "
     - "User: "
     - "Image: "
     - "Media: "
     - "Arbitration in location"
     - "Communications in location"
     - "Constitutional history of location"
     - "Economy of location"
     - "Demographics of location"
     - "Foreign relations of location"

  - "Geography of location"
  - "History of location"
  - "Military of location"
  - "Politics of location"
  - "Transport in location"
  - "Outline of topic"
- Should not match `\d\d\d\d in location` or `location in \d\d\d\d`
- Should not be names of disabiguation pages. To make this inclusive for all relevant pages, including typos, that means symbols that match `\([Dd]isambig[^)]*\)`
- Synonyms that both a) could be mistaken for ones that start with articles and b) might subsume something useful. That means that for example "A. House" (synonym of "Abraham House") is disqualified because it might mislead START in the case of questions like "How much does a house cost in the Silicon Valley?". On the other hand "a priori" can be kept because there are no sensible queries where "a" is an article before "priori".

2. Synonym generation

   To accommodate these restrictions two methods are employed. Disqualification and modification of synonym candidates. First modification is attempted and if that fails we disqualify. The rules for modification are as follows:

   - Strip determiners (articles) that are at the beginning of a synonym (or would be at the beginning if not for punctuation):
     - "A "
     - "An "
     - "The "
     - '(The) '
     - The
     - etc.
   - Generate both versions, with and without paren. Eg given symbol "Raven (journal)" generate both:
     - "Raven (journal)"
     - "Raven"

- Generate before and after slash, but not the original symbol, e.g.:
  - Given symbol "Russian language/Russian alphabet" generate both
    * "Russian language"
    * "Russian alphabet"
- Reverse inverted synonyms with commas. Eg given synonym "Congo, Democratic Republic Of The" invert it to get "Democratic Republic Of The Congo"
- As usual, get rid of leading articles if necessary. Eg given synonym "Golden ratio, the" replace it with "the Golden ratio", then strip articles to get: "Golden ratio" same goes for a, an, etc.

This way we generate an initial set of synonyms from the name of the object itself. Furthermore we can generate a set of synonyms from wikipedia redirects to the article. Wikipedia kindly provides an SQL dump for all redirects.

To load the table, in your database where you have loaded the wikipedia data, you should load the redirects table:

```
wget https://dumps.wikimedia.org/enwiki/latest/
   enwiki-latest-redirect.sql.gz \
  -O redirect.sql.gz && gzcat redirect.sql.gz |
     mysql
```

And then from the SQL db to find all (good and bad) synonyms to Bill Clinton you can:

```
mysql> select page_title, rd_title from redirect
   join page on rd_from = page_id and (rd_title =
    "Bill_Clinton" or page_title = "Bill_Clinton"
   );
+--
   ------------------------------------+--------------+

| page_title                           | rd_title
       |
+--
   ------------------------------------+--------------+
```

```
| BillClinton                          |
   Bill_Clinton |
| William_Jefferson_Clinton            |
   Bill_Clinton |
| President_Clinton                    |
   Bill_Clinton |
| William_Jefferson_Blythe_IV          |
   Bill_Clinton |
| Bill_Blythe_IV                       |
   Bill_Clinton |
| Clinton_Gore_Administration          |
   Bill_Clinton |
| Buddy_(Clinton’s_dog)␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣
   Bill_Clinton␣|
|␣Bill_clinton␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣
   Bill_Clinton␣|
|␣William_Jefferson_Blythe_III␣␣␣␣␣␣␣␣|␣
   Bill_Clinton␣|
|␣President_Bill_Clinton␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣
   Bill_Clinton␣|
|␣Bull_Clinton␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣
   Bill_Clinton␣|
|␣Clinton,_Bill␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣
   Bill_Clinton␣|
|␣William_clinton␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣
   Bill_Clinton␣|
|␣42nd_President_of_the_United_States␣|␣
   Bill_Clinton␣|
|␣Bill_Jefferson_Clinton␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣
   Bill_Clinton␣|
|␣William_J._Clinton␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣
   Bill_Clinton␣|
|␣Billl_Clinton␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣
   Bill_Clinton␣|
|␣Bill_Clinton\␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣
   Bill_Clinton␣|
|␣Bill_Clinton’s_Post_Presidency      |
   Bill_Clinton |
| Bill_Clinton’s_Post-Presidency␣␣␣␣␣␣|␣
   Bill_Clinton␣|
|␣Klin-ton␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣
   Bill_Clinton␣|
|␣Bill_J._Clinton␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣
   Bill_Clinton␣|
|␣William_Jefferson_"Bill"_Clinton␣␣␣␣|␣
```

```
   Bill_Clinton␣|
|␣William_Blythe_III␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣
   Bill_Clinton␣|
|␣William_J._Blythe␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣
   Bill_Clinton␣|
|␣William_J._Blythe_III␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣
   Bill_Clinton␣|
|␣Bil_Clinton␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣
   Bill_Clinton␣|
|␣WilliamJeffersonClinton␣␣␣␣␣␣␣␣␣␣␣␣␣|␣
   Bill_Clinton␣|
|␣William_J_Clinton␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣
   Bill_Clinton␣|
|␣Bill_Clinton's_sex_scandals         |
   Bill_Clinton |
| Billy_Clinton                       |
   Bill_Clinton |
| Willam_Jefferson_Blythe_III         |
   Bill_Clinton |
| William_"Bill"_Clinton              |
   Bill_Clinton |
| Billll_Clinton                      |
   Bill_Clinton |
| Bill_Klinton                        |
   Bill_Clinton |
| William_Clinton                     |
   Bill_Clinton |
| Willy_Clinton                       |
   Bill_Clinton |
| William_Jefferson_(Bill)_Clinton    |
   Bill_Clinton |
| Bubba_Clinton                       |
   Bill_Clinton |
| MTV_president                       |
   Bill_Clinton |
| MTV_President                       |
   Bill_Clinton |
| The_MTV_President                   |
   Bill_Clinton |
| Howard_G._Paster                    |
   Bill_Clinton |
| Clintonesque                        |
   Bill_Clinton |
| William_Clinton                     |
   Bill_Clinton |
```

```
| William_Jefferson_Clinton           |
   Bill_Clinton |
+--
   ------------------------------------+--------------+

46 rows in set (11.77 sec)
```

4.9 Backend databases

Wikipediabase uses primarily a remote data store that implements the mediawiki interface and attempts to deal with the arising performance issues by aggressively caching pages to a backend key-value based database. The interface with the database is abstracted by using a python-style dictionary interface.

1. DBM

   Several dbm implementations are provided by the python standard library. These include:

   (a) ndbm

2. SQLite

3. Redis

4. Postgres

4.10 Data sources

1. HTML

   The initial approach to getting the data is to retrieve the normal HTML versions of wikipedia articles and using edit pages to retrieve the mediawiki markup. We invariably use the original wikipedia.org site for performance reasons (See wikipedia-mirror runtime performance section).

2. API

   Mediawiki provides an API for all the required functionality.

   (a) Performance

3. Dumps / Database

   Direct interface with a local database, besides caching using mdb and/or sqlite was not implemented as part of the thesis. However shotrly after caching and compile time data pools in redis and postrgres were implemented.

4.11 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:

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

   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

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

   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

4.12 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

5 WikipediaMirror

5.1 The xerces bug

At the time of writing mwdumper a strange, semi-random bug. While `make sql-dump-parts` is running the following is encountered:

```
...

376,000 pages (14,460.426/sec), 376,000 revs
   (14,460.426/sec)
377,000 pages (14,458.848/sec), 377,000 revs
   (14,458.848/sec)
Exception in thread "main" java.lang.
   ArrayIndexOutOfBoundsException: 2048
        at org.apache.xerces.impl.io.UTF8Reader.read(
           Unknown Source)
        at org.apache.xerces.impl.XMLEntityScanner.
           load(Unknown Source)
        at org.apache.xerces.impl.XMLEntityScanner.
           scanContent(Unknown Source)
        at org.apache.xerces.impl.
           XMLDocumentFragmentScannerImpl.scanContent
           (Unknown Source)
        at org.apache.xerces.impl.
           XMLDocumentFragmentScannerImpl$FragmentContentDispatcher
           .dispatch(Unknown Source)
        at org.apache.xerces.impl.
           XMLDocumentFragmentScannerImpl.
           scanDocument(Unknown Source)
        at org.apache.xerces.parsers.
           XML11Configuration.parse(Unknown Source)
        at org.apache.xerces.parsers.
           XML11Configuration.parse(Unknown Source)
        at org.apache.xerces.parsers.XMLParser.parse(
           Unknown Source)
        at org.apache.xerces.parsers.
           AbstractSAXParser.parse(Unknown Source)
        at org.apache.xerces.jaxp.
           SAXParserImpl$JAXPSAXParser.parse(Unknown
           Source)
        at javax.xml.parsers.SAXParser.parse(
           SAXParser.java:392)
        at javax.xml.parsers.SAXParser.parse(
           SAXParser.java:195)
```

```
        at org.mediawiki.importer.XmlDumpReader.
           readDump(XmlDumpReader.java:88)
        at org.mediawiki.dumper.Dumper.main(Dumper.
           java:142)
make: *** [/scratch/cperivol/wikipedia-mirror/drafts/
   wikipedia-parts/enwiki-20131202-pages-articles20.
   xml-p011125004p013324998.sql] Error 1
```

Inspecting the makefules and running `make --just-print sql-dump-parts` we find out that the failing command is:

```
$ java -jar /scratch/cperivol/wikipedia-mirror/tools/
   mwdumper.jar   --format=sql:1.5 /scratch/cperivol/
   wikipedia-mirror/drafts/wikipedia-parts/enwiki
   -20131202-pages-articles20.xml-
   p011125004p013324998.fix.xml > /root/path/
   wikipedia-parts//enwiki-20131202-pages-articles20.
   xml-p011125004p013324998.sql
```

Fortunately this does not run for too long so we can safely experiment. Here is the `time` output:

```
26.65s user 1.73s system 78% cpu 35.949 total
```

The error seems to be during reading of the XML dump so it is not specific to SQL output. This could be useful for figuring out which article causes the error, removing which will hopefully resolve the error. To find that out we first try exporting to XML:

```
$ java -jar /scratch/cperivol/wikipedia-mirror/tools/
   mwdumper.jar   --format=xml /scratch/cperivol/
   wikipedia-mirror/drafts/wikipedia-parts/enwiki
   -20131202-pages-articles20.xml-
   p011125004p013324998.fix.xml > /tmp/just-a-copy.
   xml
```

As expected the same error as above is yielded. We then look for the last article two it tried to export by printing in reverse order the output xml file, finding the last two occurances of *<title>* with `grep` and reverse again to print them in the original order (note that `tac` is like `cat`, only that yields lines in reverse order):

```
$ tac /tmp/just-a-copy.xml | grep "<title>" -m 2 |
    tac
       <title>The roaring 20s</title>
       <title>Cranopsis bocourti</title> # <- This is
           the last one
```

This operation finishes quickly despite `/tmp/just-a-copy.xml` being fairly large because `tac` seeks to the end of the file and reads backwards until `grep` finds the 2 occurances it is looking for and quits. On ext3 the seek operation does not traverse the entire file. Indeed from the `tac` source code:

```
if (lseek (input_fd, file_pos, SEEK_SET) < 0)
    error (0, errno, _("%s: seek failed"), quotef (
       file));
/* Shift the pending record data right to make room
   for the new.
   The source and destination regions probably
      overlap.  */
memmove (G_buffer + read_size, G_buffer,
   saved_record_size);
past_end = G_buffer + read_size + saved_record_size;
/* For non-regexp searches, avoid unnecessary
   scanning. */
if (sentinel_length)
    match_start = G_buffer + read_size;
else
    match_start = past_end;

if (safe_read (input_fd, G_buffer, read_size) !=
   read_size)
{
    error (0, errno, _("%s: read error"), quotef (
       file));
    return false;
}
```

Let's save the path of the original xml file in a variable as we will be using it a lot. So from now on `$ORIGINAL_XML` will be the path of the original xml.

```
$ export ORIGINAL_XML=/scratch/cperivol/wikipedia-
   mirror/drafts/wikipedia-parts/enwiki-20131202-
   pages-articles20.xml-p011125004p013324998.fix.xml
```

First let's see if there is anything strange going on in the xml file:

```
$ grep "<title>Cranopsis␣bocourti</title>" -A 200 -B
   100 $ORIGINAL_XML | less
```

`| less` is to browse and `-A 200 -B 100` means *"show 200 lines after and 100 before the matching line"*. Nothing peculiar was found, so we can't really fix the problem in-place, we will try crudely removing the entire article and hope it works (spoiler alert: it does).

We will try to inspect the parents of the `title` of the breaking article. Fortunately the generated xml is indented so we can find the parents based on that. We count 6 spaces of indentation so we will search backwards from there on each level of indentation. The first line we find on each case will be a direct parent of the article.

```
$ for i in {0..6}; do \
    echo "Level␣$i:"; \
    tac /tmp/just-a-copy.xml | grep "^␣\{$i\}<[^/]" -
       m 1 -n | tac; \
done

Level 0:
17564960:<mediawiki xmlns="http://www.mediawiki.org/
   xml/export-0.3/" xmlns:xsi="http://www.w3.org
   /2001/XMLSchema-instance" xsi:schemaLocation="http
   ://www.mediawiki.org/xml/export-0.3/␣http://www.
   mediawiki.org/xml/export-0.3.xsd" version="0.3"
   xml:lang="en">
Level 1:
Level 2:
38:  <page>
Level 3:
Level 4:
35:    <revision>
Level 5:
Level 6:
26:      <text xml:space="preserve">&lt;!-- This
   article was auto-generated by [[User:Polbot]]. --&
   gt;
```

Looks like the xml is just `page` s thrown in a grand domain called `mediawiki`. We could have seen that from the java source too but as expensive as this is, it is much faster than dealing with the source of `mwdumper`.

The easiest way to cut off this article would be `awk` but that will take ages and we want to optimize and automate this entire process. First let's try just plain comparing the articles:

```
$ cmp /tmp/just-a-copy.xml $ORIGINAL_XML
/tmp/just-a-copy.xml /scratch/cperivol/wikipedia-
   mirror/drafts/wikipedia-parts/enwiki-20131202-
   pages-articles20.xml-p011125004p013324998.fix.xml
   differ: byte 2, line 1
```

That was fast... Let's see what went wrong:

```
$ head $ORIGINAL_XML
<mediawiki xmlns="http://www.mediawiki.org/xml/export
   -0.8/" xmlns:xsi="http://www.w3.org/2001/XMLSchema
   -instance" xsi:schemaLocation="http://www.
   mediawiki.org/xml/export-0.8/␣http://www.mediawiki
   .org/xml/export-0.8.xsd" version="0.8" xml:lang="
   en">
  <siteinfo>
    <sitename>Wikipedia</sitename>
    <base>http://en.wikipedia.org/wiki/Main_Page</
       base>
    <generator>MediaWiki 1.23wmf4</generator>
    <case>first-letter</case>
    <namespaces>
      <namespace key="-2" case="first-letter">Media</
         namespace>
      <namespace key="-1" case="first-letter">Special
         </namespace>
      <namespace key="0" case="first-letter" />

$ head /tmp/just-a-copy.xml
<?xml version="1.0" encoding="utf-8" ?>
<mediawiki xmlns="http://www.mediawiki.org/xml/export
   -0.3/" xmlns:xsi="http://www.w3.org/2001/XMLSchema
   -instance" xsi:schemaLocation="http://www.
   mediawiki.org/xml/export-0.3/␣http://www.mediawiki
   .org/xml/export-0.3.xsd" version="0.3" xml:lang="
   en">
  <siteinfo>
    <sitename>Wikipedia</sitename>
    <base>http://en.wikipedia.org/wiki/Main_Page</
       base>
```

```
<generator>MediaWiki 1.23wmf4</generator>
<case>first-letter</case>
<namespaces>
  <namespace key="-2">Media</namespace>
```

The attributes of the xml tags are quite different. Our best chance is if the line numbers match up. We count the numbers of lines in `/tmp/just-a-copy.xml` and hope that the corresponding line number in `$ORIGINAL_XML` will be the same line. If that is so we can ignore the the contextual xml information and just blank out the problematic article. We will use `wc` which is also quite fast.

```
$ wc -l /tmp/just-a-copy.xml
17564961 /tmp/just-a-copy.xml
```

And the corresponding line in `$ORIGINAL_XML` would be about:

```
$ sed "17564960q;d" $ORIGINAL_XML
[[Willie Jones (American football)|Willie Jones]],
```

Football... nothing to do with frogs. Looks like there is no avoiding some level of parsing.

1. Parsing

   We will make the following assumptions to avoid properly parsing the document:

   - The XML in the original file is valid
   - Any XML within the articles is HTML escaped

   First off working with lines is slow because user space code needs to look for newlines. Working bytes delegates work to the kernel, speeding things up considerably. So the `dd` is the right tool for the job. So we will first find at which byte is the article I am interested in.

   ```
   $ grep -b "<title>Cranopsis␣bocourti</title>" -m
      1 $ORIGINAL_XML
   1197420547:    <title>Cranopsis bocourti</title>
   ```

This may take a little while but you are stuck with it unfortunately. Our stategy is to make two files: `/tmp/original_tail.xml` that will contain all the data *after* the page we want to remove and `/tmp/original_head.xml` that will contain all the data *before* the page we want to remove.

Now we will use `sed` to look for `</page>` after byte 1197420547 which will be point x we will and dump the contents of `$ORIGINAL_XML` after point x:

```
$ dd if=$ORIGINAL_XML skip=1197420547 ibs=1 | sed
    '0,/<\/page>/d' > /tmp/original_tail.xml
```

Great, that worked! `dd` does not copy in reverse so we will need to do something more complex to construct `/tmo/original_head.xml`. Let's say the position where we found the title of the page we want to remove is $\alpha = 1197420547$ and the point where the page starts is point β. It is fairly safe to assume that $\beta > \alpha - 1000$ (we can calibrate the constant 1000 if that assumption is wrong, but it turns out that it isn't). This way we only need to search into 1Kb for `<page>`. Effectively instead of copying the bytes in range $[0, \beta)$ we are concatenating two ranges $[0, \alpha - 1000] \cup (\alpha - 1000, \beta)$ by making a subshell that will first output the first range and then output $(\alpha - 1000, \alpha)$ stopping when it finds `<page>`. Here is the one liner:

```
$ (dd count=$((1197420547-1000)) ibs=1 if=
   $ORIGINAL_XML; \
   dd if=$ORIGINAL_XML count=1000 skip=$
      ((1197420547-1000)) ibs=1 \
       | tac | sed '/<page>/,$d' | tac) > /tmp/
          original_head.xml
```

2. The final solution

   All the above was used to compose a script that lives in `data/xml-parse.sh` which is utilised by the makefiles to remove all problematic articles. If mwdumper fails, we identify the article that caused the breakage and remove it using `xml-parse.sh`. Then we rerun mwdumper. We repeat that until mwdumper succeeds. In total the conflicting articles are aboit 10-15, and are different depending on the dump being used.

3. Covering up with spaces

From the above exploration of ways for circumventing the issue of the breaking article we omitted a fairly obvious, but thematically different approach: covering up breaking article with spaces. Once we find out the range in which the page resides we can `mmap` precisely in that part of `$ORIGINAL_XML` and then `memset` covering it up with space characters. The actual implementation lives in `data/page_remover.c`, below we present the call to mmap:

```
ctx->off = off-pa_off;
ctx->fd = open(fname, O_RDWR, 0x0666);
if (ctx->fd == -1) {
    perror("open");
    return NULL;
}

ctx->size = len;
ctx->data = mmap(0, len+ctx->off, PROT_READ |
   PROT_WRITE,
                MAP_SHARED, ctx->fd, pa_off);
if (ctx->data == MAP_FAILED) {
    perror ("mmap");
    return NULL;
}
```

and the `mmemset`:

```
/* You MIGHT want to thread this but I dont think
    it will make
 * much more difference than memset. */
memset(ctx->data + ctx->off, '␣', ctx->size);
```

Surprisingly this did not fix the mwdumper issue which points to a possible memory leak on the part of xerces but it is beyond the scope of this project to debug fix that if we have a choice.

4. The sed command

   Above we kind of glazed over our use the `sed` command but it might be interesting to spend some ink on it. Sed is a unix tool found in coreutils that according to it's man page is a

   > stream editor for filtering and transforming text.

The basic premise is that the *"pattern space"*, or the input stream which is a normal unix stream coming from a file, a pipe or just stdin, is passed through a programmable pipeline. Either the modified pattern space itself is printed or, with the use of the `-n` flag, selected parts of it. Let's look at the use that we have made for sed above

Initially we used sed to print a specific line in a file:

```
$ sed "17564960q;d"
```

This sed program is separated by a semicolon. Sed iterates over the lines of the input stream and runs each of the `;` separated commands on them in sequence until one succeeds. The commands here are `17564960q` and `d`. `17564960q` will quit sed once line 17564960 is reached. `d` will discard the current line. So sed discards lines until it reaches line 17564960 which it prints and quits.

We then used a sed command as part of a series of shell commands piped together in order to print all the lines of a stream after a specific pattern (in our case `</page>`).

```
$ sed '0,/<\/page>/d'
```

This time we have only a single sed command, `d`. Sed iterates over the lines in the stream, discarding lines in the range of lines 0 to the line that matches `<\/page>`, effectively only printing lines after `</page>`.

Our final use of sed is the inverse of the aforementioned one,

```
$ sed '/<page>/,$d'
```

Here sed iterates again over all the lines of the stream this time discarding lines in the range between the first line that matches `<page>` until the final line, denoted with a `$`.

5.2 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) $\text{page}_{\text{remover.c}}$

As previously discussed, the `xerces` library that `mwdumper` depends on fails, seemingly at random, to process certain pages. To address this issue we remove the pages completely and retry. Since this task is fairly straight forward yet performance sensitive we resorted to writing a small low level program in C to address it, `page_remove.c`. Page remover accepts as input the path of the XML wikipedia dump, the offset of the article and the size of the article. It then uses the `mmap` system call to random-access the data within the file and fill the article with withespace characters. `page_remover.c` is not threaded as the bottleneck is the HDD IO speed.

(b) sql-clear.sh

`sql-clear.sh` is a small bash script that truncates all tables from a database. Truncating means leaving the table scheamata unaffected and delete all internal data.

(c) utf8thread.c

`utf8thread.c` is another low level program that blanks out all invalid utf-8 characters from a file. We used `pthreads` to speed things up.

(d) webmonitor.py

`webmonitor.py` is a python script that sets up a web page that shows live data in the form of a histogram about the progress of the database population. `webmonitor.py` serves a static html page and then deeds it the data over websocket. Webmonitor can show any stream of `<epoc date> <float value>` pairs that it receives in it's input. As a sample:

```
$ pip install tornado
```

First install the dependencies of the script. That would be tornado, an asynchronous web framework supporting websockets. We will instruct `tornado` tornado will serve the following page:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD␣HTML␣
   4.01//EN" "http://www.w3.org/TR/html4/
   strict.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="
       text/html;␣charset=utf-8">
    <title>DrNinjaBatmans Websockets</title>

    <script type="text/javascript" src="http
       ://code.jquery.com/jquery-1.10.1.js"><
       /script>
    <script type="text/javascript" src="http
       ://code.highcharts.com/highcharts.js">
       </script>

    <script>
     var chart; // global
     var url = location.hostname + ':' + (
        parseInt(location.port));
     var ws = new WebSocket('ws://' + url +
        '/websocket');
     ws.onmessage = function(msg) {
         add_point(msg.data);
     };

     // ws.onclose = function() { alert('
        Connection closed.'); };
```

```
        var add_point = function(point) {
            var series = chart.series[0],
            shift = series.data.length > %d;

            chart.series[0].addPoint(eval(point)
                , true, shift);
        };

        $(document).ready(function() {
            chart = new Highcharts.Chart(JSON.
                parse('%s'));
        });
       </script>

    </head>
    <body>
        <div id="container" style="width:␣800px
            ;␣height:␣400px;␣margin:␣0␣auto"></
            div>
    </body>
</html>
```

In essence this page expects to read a stream of values from a websocket at `ws://localhost:8888/hostname` – although it is smart enough to change the `localhost:8888` if you are serving this to another location – and plot them in real time using `highcharts.js`.

The attentive reader may notice that the above is not quite HTML but rather a python formatted string. That is for two reasones. First because the script handles the configuration (see `chart = new Highcharts.Chart(JSON.parse('%s'));`). Second because the width of the graph will is calulated at page load time and the plot needs to be shifted to only show the most recent points.

```
$ for i in {1..100}; do echo $i;  sleep  1;
    done | \
     awk -oL "{print␣\$1/100}" | \
     python webmonitor.py
```

This will produce, in 1 second intervals, numbers from 1 to 100. Then it normalizes them using `awk` and feeds them to webmonitor. After this command executes we can open the browser and then

navigate to `localhost:8888`.

We utilize this to remotely monitor the total size of data that `mysql` consumes.

(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
```

5.3 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.

5.4 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

5.5 Automation

Creating a wikipedia mirror may seem like a straight forward task but it involves many caveats, nuances and repetitive tasks. Multiple methods of autmoation were employed to carry out the wide variety of tasks involved into the process.

1. Makefiles / laziness

   The most important part of wikipedia-mirror automation is the `make` build system. Make is a build system whereby one can declare required files (targets), dependencies for them, and a set of shell commands that will build those targets. For example, save the following as `Makefile` in a project that contains the files `foo.c`, `foo.h`, `bar.c` and `bar.h`:

   ```
   foo.o: foo.c foo.h
        gcc foo.c -c -o foo.o

   bar.o: bar.c
       gcc bar.c -c -o bar.o

   foobar: foo.o bar.o
        gcc foo.o bar.o -o foobar
   ```

   this means that to build `foobar` we need `foo.o` and `bar.o`. And to build `foo.o` and `bar.o` we need `foo.c` and `foo.h`, and `bar.c` and `bar.h` respectively. We also provide commands for building `foo.o`, `bar.o` and `foobar`, which are

   - `gcc foo.c -c -o foo.o`
   - `gcc bar.c -c -o bar.o`
   - and `gcc foo.o bar.o -o foobar`

   respectively. Notice that there are no rules for the `.c` and `.h` files. That is because `make` should fail if they are not present. So if we run `make foobar`, make will check for `foobar`'s existence and modification date. If foobar is missing or it's modification date is earlier than it's deopendecies' (ie `foo.o` and `bar.o`) it will be rebuilt. If any dependecies are missing the same logic is applied to that. This way if we build foobar once, and then edit `bar.c` and rerun `make foobar`, `make` will recursively deduce that

- `bar.o` is out of date with respect to it's dependency `bar.c`
- When `bar.o` is rebuilt it now has a more recent modification date than `foobar` and therefore the latter is out of date with respect to it's dependency `bar.o` so it needs to be rebuilt.

This way make can infer a near optimal strategy for building each time the minimum amount of required targets.

Now that we made the basic logic of `make` clear let's dive into some basic features that make our life easier.

(a) Phony targets

Some tasks do not result in a file and thay need to be run every time `make` encounters them in the dependency tree. For this we have the special keywork `.PHONY:`. Here is an example.

```
.PHONY:
clean:
    rm -rf *
```

This tells make that no file named `clean` will emerge from running `rm -rf *`, and also that even if an up-to-date file named `clean` exists, this target is to be run regardless.

It is worth noting that phony dependencies will always render the dependent target out-of-date. For example:

```
.PHONY:
say-hello:
    echo "hello"

test.txt: say-hello
    touch test.txt
```

When `touch test.txt` will be run **every** time we run `make test.txt` simply becaus make can not be sure that the phony target `say-hello` did not change anything important for `test.txt`. For this reason phony targets are only meant for user facing tasks.

(b) Variables

`makefiles` can have variables defined in a variety of ways. The most basic case is the following

```
OBJCETS = foo.o bar.o

show:
    echo $(OBJECTS)
```

Running `make show` wil print `foo.o bar.o` to the console.

2. Shell scripts
3. Bitnami

5.6 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.

5.7 Appendix (script sources)

1. page$_{\text{remover.c}}$

```
/*
 * Copyright 2014 Chris Perivolaropoulos <
    cperivol@csail.mit.edu>
 *
 * This program is free software: you can
    redistribute it and/or
 * modify it under the terms of the GNU General
    Public License as
 * published by the Free Software Foundation,
    either version 3 of the
 * License, or (at your option) any later version
    .
 *
 * This program is distributed in the hope that
    it will be useful, but
 * WITHOUT ANY WARRANTY; without even the implied
     warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR
    PURPOSE.
 *
 * See the GNU General Public License for more
    details.  You should
 * have received a copy of the GNU General Public
     License along with
 * this program.
 *
 * If not, see <http://www.gnu.org/licenses/>.
 *
 * This should fill a range in a file with spaces
    . This is an in-place
 * operation so it should be pretty fast.
 *
 * Usage: page_remover PATH OFFSET LENGHT
 */

#include <assert.h>
#include <fcntl.h>
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
#include <sys/mman.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <semaphore.h>
#include <unistd.h>
#include <unistd.h>

#define USAGE_INFO "page_remover␣PATH␣OFFSET␣
   LENGTH"
#define PRINT(ctx, args...) do { sem_wait(&ctx->
   stdio_mutex);   \
                                printf(args);
                                           \
                                fflush(stdout);
                                           \
                                sem_post(&ctx->
                                   stdio_mutex);
                                        \
                                } while(0)

typedef struct context {
    int fd;
    size_t size;
    off_t off;
    sem_t stdio_mutex;
    void* data;
} context_t;

context_t* context_init(char* fname, off_t off,
   size_t len)
{
    context_t * ctx = (context_t*)malloc(sizeof(
       context_t));
    off_t pa_off = off & ~(sysconf(_SC_PAGE_SIZE)
        - 1);

    sem_init(&ctx->stdio_mutex, 0 /* Shared.
       Usually ignored */ , 1);

    PRINT(ctx, "Opening␣%s␣at␣%lu␣(len:␣%lu)\n",
       fname, off, len);

    ctx->off = off-pa_off;
    ctx->fd = open(fname, O_RDWR, 0x0666);
```

```
    if (ctx->fd == -1) {
      perror("open");
      return NULL;
    }

    ctx->size = len;
    ctx->data = mmap(0, len+ctx->off, PROT_READ |
        PROT_WRITE,
                     MAP_SHARED, ctx->fd, pa_off);
    if (ctx->data == MAP_FAILED) {
      perror ("mmap");
      return NULL;
    }

    return ctx;
}

void context_destroy(context_t* ctx)
{
    if (close (ctx->fd) == -1)
      perror ("close");

    if (munmap ((void*)ctx->data, ctx->size) ==
       -1)
      perror ("munmap");

    sem_destroy(&ctx->stdio_mutex);
    free(ctx);
}

int main(int argc, char *argv[])
{
    if (argc != 4)
      fprintf(stderr, USAGE_INFO);

    context_t *ctx = context_init(argv[1], atoi(
       argv[2]), atoi(argv[3]));

    /* You MIGHT want to thread this but I dont
       think it will make
     * much more difference than memset. */
    memset(ctx->data + ctx->off, '␣', ctx->size);

    context_destroy(ctx);
    return 0;
```

```
}
```

2. utf8thread.c

```
#include <assert.h>
#include <fcntl.h>
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <semaphore.h>
#include <unistd.h>
#include <unistd.h>

sem_t stdio_mutex;

#define PRINT(args...) do {sem_wait(&stdio_mutex)
   ;      \
      printf(args);
                \
      fflush(stdout);
                \
      sem_post(&stdio_mutex);
                \
    } while(0)

/* #define DEBUG(args...)                PRINT(
   args) */
#define DEBUG(...)

#define DEFAULT_CHAR '␣'
#define WORKERS 8
#define MESSAGE_DENSITY 1000000000

typedef unsigned long long u64;

#define UTF_LC(l) ((0xff >> (8 - (l))) << (8 - (l
   )))
#define UTF_CHECK(l, c) (((UTF_LC(l) & (c)) ==
   UTF_LC(l)) && (0 == ((c) & (1 << (7-(l))))))
```

```
#define UTF_LEN(x) (UTF_CHECK(6, x) ? 6 :       \
                    UTF_CHECK(5, x) ? 5 : \
                    UTF_CHECK(4, x) ? 4 : \
                    UTF_CHECK(3, x) ? 3 : \
                    UTF_CHECK(2, x) ? 2 : -1)

struct crange {
    u64 start, end;
};

/* Get return the next character after the last
   correct one. */
inline u64 valid_utf8(u64 c)
{
    char i;
    /* Ascii */
    if ((*(char*)c & 0x80) == 0)
      return c+1;

    /*  */
    for (i = UTF_LEN(*(char*)c)-1; i>0; i--) {
      c++;
      if (!UTF_CHECK(1, *(char*)c)) {
          return (u64)NULL;
      }
    }

    return i<0 ? 0 : c+1;
}


void* fix_range(void* _r)
{
    struct crange* r = _r;
    u64 tmp, id = r->start;
    long long unsigned count = 0;

    while ((u64)r->start < (u64)r->end) {
      if (count++ % MESSAGE_DENSITY == 0)
          printf ("[worker:␣0x%016llx]␣Done␣with␣
             %lluK.\n", id, count % 1024);

      if (!(tmp = valid_utf8(r->start))){
          PRINT("Invalid␣char␣0x%x␣(next:␣0x%x)\n
```

```
            ",
               *(char*)r->start, *(char*)(r->
                  start+1));
            *((char*)r->start) = DEFAULT_CHAR;
            (r->start)++;
        } else {
            r->start = tmp;
        }
    }

    PRINT ("[worker:␣0x%016llx]␣OUT\n", id);
    return NULL;
}

void run(u64 p, u64 sz)
{
    int n, i;
    u64 wsize;
    pthread_t workers[WORKERS];
    struct crange rngs[WORKERS];

    wsize = sz/WORKERS + 1;
    printf("Base␣address:␣0x%016llx,␣step␣size:␣0
       x%016llx\n", p, wsize);

    for (i=0; i<WORKERS; i++){
      rngs[i].start = p + wsize*i;
      rngs[i].end = p + wsize*i + wsize;

      PRINT("Spawning␣worker␣%d␣on␣range␣[0x%016
         llx,␣0x%016llx),␣%llu␣bytes...", i, rngs
         [i].start, rngs[i].end, wsize);
      if ((n = pthread_create(workers+i, NULL,
         fix_range, (void*)(rngs+i)))) {
          PRINT("FAIL\n");
          perror("worker");
          return;
      }
      PRINT("OK\n");
    }

    PRINT ("Wrapping␣up...\n");
    for (i=0; i<WORKERS; i++) {
      PRINT ("Joining␣worker␣%d...", i);
      pthread_join(workers[i], NULL);
```

```
      PRINT ("OK\n");
      PRINT("Worker␣%d␣went␣through␣%llu␣bytes.\n
          ",
            i, (u64)rngs[i].end - (u64)rngs[i].
               start);
    }
}


int main(int argc, char *argv[])
{
    int fd;
    long long int sz, p;
    struct stat buf;

    sem_init(&stdio_mutex, 0 /* Shared. Usually
       ignored */ , 1);

    fd = open(argv[1], O_RDWR, 0x0666);
    if (fd == -1) {
      perror("open");
      return 1;
    }

    fstat(fd, &buf);
    sz = buf.st_size;
    printf("File␣size:␣0x%016llx\n", sz);

    p = (u64)mmap (0, buf.st_size, PROT_READ |
       PROT_WRITE , MAP_SHARED, fd, 0);
    if (p == -1) {
      perror ("mmap");
      return 1;
    }

    run(p, buf.st_size);

    if (close (fd) == -1) {
      perror ("close");
      return 1;
    }


    if (munmap ((void*)p, buf.st_size) == -1) {
      perror ("munmap");
```

```
        return 1;
    }

    sem_destroy(&stdio_mutex);

    return 0;
}
```

3. sql-clear.sh

```
#!/bin/bash
MUSER="$1"
MPASS="$2"
MDB="$3"
MYSQL=$4

# Detect paths
AWK=$(which awk)
GREP=$(which grep)

if [ $# -ne 4 ]
then
      echo "Usage: $0 {MySQL-User-Name} {MySQL-
         User-Password} {MySQL-Database-Name} {
         MySQL executable to use}"
      echo "Drops all tables from a MySQL"
      exit 1
fi

TABLES=$($MYSQL -u $MUSER -p$MPASS $MDB -e 'show
   tables' | $AWK '{ print $1}' | $GREP -v '^
   Tables' )

for t in $TABLES
do
      echo "Clearing $t table from $MDB database
         ..."
      $MYSQL -u $MUSER -p$MPASS $MDB -e "truncate
         table $t"
done
```

4. webmonitor.py

```
"""
Just␣feed␣pairs␣of

<epoc␣date>␣<float␣value>

or␣even␣just

<float␣value>

One␣way␣to␣do␣that␣would␣be

␣␣␣␣$␣<cmd>␣stdbuf␣-oL␣awk␣"{print␣\$1/$$max}"␣|␣
    python␣webmonitor.py

and␣I␣will␣plot␣them␣on␣port␣8888.␣This␣will␣also
    ␣pipe␣the␣input␣right
out␣to␣the␣output.␣Strange␣input␣will␣be␣ignored␣
    and␣piped␣this␣way,
but␣this␣needs␣to␣be␣done␣by␣awk␣aswell␣in␣the␣
    above␣example.
"""

import sys
import json
import time

from threading import Thread
from collections import deque

import tornado.websocket as websocket
import tornado.ioloop
import tornado.web

HTML = """
<!DOCTYPE␣HTML␣PUBLIC␣"-//W3C//DTD␣HTML␣4.01//EN"
    ␣"http://www.w3.org/TR/html4/strict.dtd">
<html>
␣␣<head>
␣␣␣␣<meta␣http-equiv="Content-Type"␣content="text
    /html;␣charset=utf-8">
␣␣␣␣<title>DrNinjaBatmans␣Websockets</title>

␣␣␣␣<script␣type="text/javascript"␣src="http://
    code.jquery.com/jquery-1.10.1.js"></script>
```

```
    <script type="text/javascript" src="http://
        code.highcharts.com/highcharts.js"></script>

    <script>
var chart; // global
var url = location.hostname + ':' + (parseInt(
    location.port));
var ws = new WebSocket('ws://' + url + '/
    websocket');
ws.onmessage = function(msg) {
    add_point(msg.data);
};

// ws.onclose = function() { alert('Connection
    closed.'); };

var add_point = function(point) {
    var series = chart.series[0],
        shift = series.data.length > %d;
    chart.series[0].addPoint(eval(point), true,
        shift);
};

$(document).ready(function() {
    chart = new Highcharts.Chart(JSON.parse('%s')
    );
});
    </script>

  </head><body><div id="container" style="width:
    800px; height: 400px; margin: 0 auto"></div></
    body></html>
"""

config = {
    'visible_points': 10,
    'py_chart_opts': { 'chart': { 'renderTo': '
        container',
                                  ,
                                      defaultSeriesType
                                      ': 'spline'
                                      },
                       'title': { 'text': '
                           DrNinjaBatmans data'},
                       'xAxis': { 'type': '
```

```
                                datetime',
                                        '
                                           tickPixelInterval
                                           ': '150'},
                                'yAxis': { 'minPadding':
                                   0.2,
                                           'maxPadding':
                                              0.2,
                                           'title': {'text
                                              ': 'Value',
                                                    '
                                                       margin
                                                       ':

                                                       80}

                                       },
                                'series': [{ 'name': 'Data
                                   ',
                                             'data': []}]}

}

def date_float(s):
    try:
        date, val = s.split()
    except ValueError:
        val = s.strip()
        date = time.time()

    return int(date), float(val)


def send_stdin(fn=date_float):
    for raw in sys.stdin:
        sys.stdout.write(raw)

        # Ignore strange input.
        try:
            jsn = json.dumps(fn(raw))

            buf.append(jsn)

            for w in websockets:
                try:
```

```
                    w.write_message(jsn)
                except websocket.
                    WebSocketClosedError:
                    pass

        except:
            pass

    for ws in websockets:
        ws.close()

class StdinSocket(websocket.WebSocketHandler):
    def open(self):
        for i in buf:
            self.write_message(i)

        websockets.append(self)


    def closs(self):
        websockets.remove(self)

class MainHandler(tornado.web.RequestHandler):
    def get(self):
        self.write(HTML % (int(config['
            visible_points']),
                           json.dumps(config['
                               py_chart_opts'])))


if __name__ == "__main__":
    application = tornado.web.Application([
        (r"/", MainHandler),
        (r'/websocket', StdinSocket),
    ])
    buf = deque(maxlen=int(config['visible_points
        ']))
    websockets = []


    config['args'] = []
    for a in sys.argv[1:]:
        if '=' in a:
            k, v = a.split('=', 1)
            config[k] = v
```

```
        else:
            config['args'].append(a)

    Thread(target=send_stdin).start()
    application.listen(8888)
    tornado.ioloop.IOLoop.instance().start()
```

5. xml-parse.sh

```
  #!/bin/bash
#
# Simply removing specific articles fixes the
    xerces error with
# UTF8. If the articles are alone the error goes
    away
# aswell. Extremely weird but that's life.
    Fortunately the article is
# just a stub about some toad (Cranopsis bocourti
    )
#
# xml-parse.sh ORIGINAL_XML
    TITLE_OF_ARTICLE_TO_REMOVE [inplace]
#
# if 'inplace' is there the c program will be
    used to cover the article
# with spaces. This is much faster. Should be
    anyway. Otherwise the
# page is just ommited and the result is dumped
    in stdout. Helping
# messages are dumped in stderr After this you
    can run:
#
# java -jar tools/mwdumper.jar RESULTING_XML --
    format=sql:1.5 > SQL_DUMP

set -e
set -o pipefail

if [[ $# -lt 2 ]]; then
    echo "xml-parse.sh␣ORIGINAL_XML␣
       TITLE_OF_ARTICLE_TO_REMOVE␣[inplace]" 1>&2
    exit 0
fi
```

```
function my_dd {
    coreutils_version=$(dd --version | head -1 |
       cut -d\  -f3 | colrm 2 2 )
    if [[ $coreutils_version -ge 822 ]]; then
      eval "dd␣iflag=count_bytes␣iflag=direct␣
         oflag=seek_bytes␣ibs=1M␣$@"
    else
      echo "Your␣coreutils␣may␣be␣a␣bit␣old␣(
         $coreutils_version).␣822␣is␣the␣one␣cool
         ␣kids␣use." >&2
      eval "dd␣$@␣ibs=1"
    fi
}

ORIGINAL_XML=$1

# Dump a part of the file in sdout using dd.
# Usage:
# file_range <filename> <first_byte> <start|end|
    length>
#
# Length can be negative
function file_range {
    file=$1
    start=$2
    len=$3


    case $len in
      "end") my_dd if=$file skip=$start || exit
         1; return 0;;
      "start") my_dd if=$file count=$start ||
         exit 1; return 0;;
      "") echo "len␣was␣empty␣(file:␣$file,␣start
         :␣$start,␣len␣$len).␣Correct␣format␣<
         filename>␣<byte␣start>␣<length|'start'|'
         end'>" 1>&2; exit 1;;
      *) ;;
    esac

    if [[ $len -gt 0 ]]; then
      # Dump to stdout
      my_dd if=$file skip=$start count=$len ||
         exit 1
    else
```

```
      skip=$(($start + ($len)))
      len=$((- ($len)))

      if [[ $skip -lt 0 ]]; then
          skip=0
          len=$start
      fi

      # Dump to stdout
        my_dd if=$file skip=$skip count=$len ||
            exit 1
    fi
}

function backwards {
    tac -b | rev
}

function byte_offset {
    grep -b -o -m 1 -F  "$1" | cut -d : -f1
}

# Throw everything but the page in stdout
#
# neg_xml_page "Barack Obama"
function neg_xml_page {
    term="<title>$1</title>"
    title_offset=$(cat $ORIGINAL_XML |
       byte_offset "$term")
    echo -e "\n\tMethod:␣$2(blank␣is␣ok)" 1>&2
    echo -e "\tsearch␣term:␣$term" 1>&2
    echo -e "\tfile:␣$ORIGINAL_XML" 1>&2
    echo -e "\ttitle␣offset:␣$title_offset" 1>&2

    # Fail the term is invalid
    if [ -z "$title_offset" ]; then
      echo "Found␣'$title_offset'␣Grep-ing␣(cat␣␣
         $ORIGINAL_XML␣|␣grep␣-b␣-m␣1␣-F␣\"$term\
         "␣|␣cut␣-d:␣-f1)" 1>&2
      exit 1
    fi

    to_page_start=$(($(file_range $ORIGINAL_XML
       $title_offset -1000 | backwards |
       byte_offset "$(echo␣'<page>'␣|␣rev)")+7))
```

```
    echo -e "\tto␣page␣start␣(relative):␣
       $to_page_start" 1>&2

    file_range $ORIGINAL_XML $title_offset end |
       byte_offset "</page>" >&2
    echo $(($(file_range $ORIGINAL_XML
       $title_offset end | byte_offset "</page>")
       +7)) >&2
    to_page_end=$(($(file_range $ORIGINAL_XML
       $title_offset end | byte_offset "</page>")
       +7)) # len('</page>') == 7
    echo -e "\tto␣page␣end␣(relative):␣
       $to_page_end" 1>&2

    page_start=$(($title_offset - $to_page_start
       +1 ))
    echo -e "\tpage␣start:␣$page_start" 1>&2

    page_end=$(($title_offset + $to_page_end))
    echo -e "\tpage␣end:␣$page_end" 1>&2

    echo -e "\tbytes␣to␣copy:␣$(($(du␣-b␣
       $ORIGINAL_XML␣|␣cut␣-f1)␣-␣$page_start␣+␣
       $page_end))" 1>&2

    echo "Going␣to␣copy␣$page_start␣bytes" 1>&2
    file_range $ORIGINAL_XML $page_start start
    echo "Finished␣the␣first␣half␣up␣to␣
       $page_start,␣$((␣$(du␣-b␣$ORIGINAL_XML␣|␣
       cut␣-f␣1)␣-␣$page_end␣))␣to␣go" 1>&2
    file_range $ORIGINAL_XML $page_end end
    echo "Finished␣the␣whole␣thing." 1>&2
}

# Put stdin betwinn mediawiki tags and into
   stdout
function mediawiki_xml {
    (head -1 $ORIGINAL_XML; sed -n "/<siteinfo
       >/,/<\/siteinfo>/p;/<\/siteinfo>/q"
       $ORIGINAL_XML ; cat - ; tail -1
       $ORIGINAL_XML )
}

# 1: XML File
# 2: Article
```

```
# 3: Method (leave blank)
# Assert that the file is there and is not empty
fsize=$(du -b $ORIGINAL_XML | cut -f1)
if [[ 0 -eq $fsize ]]; then
    echo "ERROR: empty xml file $ORIGINAL_XML"
       1>&2
    exit 1
fi

echo "Will remove article '$2' from file $1 (size
   : $fsize)" 1>&2
if ! neg_xml_page "$2" "$3"; then
    ret=$?
    echo "XML parsing script failed" 1>&2
    exit $ret;
fi
```

6 Related CSAIL projects

7 Conclusion