Article from

Predictive Analytics & Futurism

December 2018

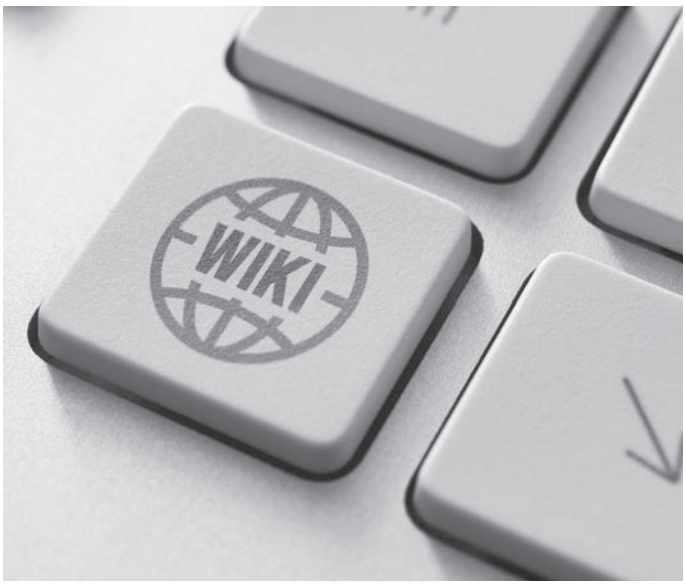
Issue 19

Extracting Medical Data From Wikipedia

By Alexandru L. Andrei

Information that includes financial, medical, demographic data, as well as facts about natural disasters is of great interest to the insurance industry. As a result, major companies, including IBM, have trained their natural language processing models using Wikipedia's dataset, one of the most comprehensive sources of information on the Internet. Any specific piece of information can be extracted from such a massive database, but the main challenge lies in processing such an enormous dataset.

Wikipedia's data is written in an XML (Extensible Markup Language) format and is presented in a manner that is both human and machine readable. However, using Wikipedia API to extract useful information is impractical due to time and traffic constraints. The file used in this article **enwiki-latest-pages-articles.xml** [1] is approximately 12.8 GB when compressed and 68 GB when decompressed, thus the majority of text editors are not able to open it. This paper describes techniques that can be used to access any specific piece of information in Wikipedia, in particular medical information, by using Wikipedia's most up-to-date dump file.



EXPLORING WIKIPEDIA DATA

The Wikipedia file is coded in the following format:

```
<mediawiki xmlns="http://www.mediawiki.org/xml/
export-0.10/" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xsi:schemaLocation="http://www.
mediawiki.org/xml/export-0.10/ http://www.mrt-0.10.
xsd" version="0.10" xml:lang="en">
  <siteinfo>
    <sitename>Wikipedia</sitename>
    <dbname>enwiki</dbname>
    <base>https://en.wikipedia.org/wiki/Main_Page</
     base>
    <generator>MediaWiki 1.32.0-wmf.14</generator>
    <case>first-letter</case>
    <namespaces>
      <namespace key="-2" case="first-letter">Media</
      namespace>
      ...
      <namespace key="2303" case="case-sensitive">
      Gadget definition talk</namespace>
    </namespaces>
  </siteinfo>
. . .
  <page>
      <page>
    <title>Anarchism</title>
    <ns>0</ns>
    <id>12</id>
    <revision>
      <id>851684166</id>
      <parentid>851684072</parentid>
      <timestamp>2018-07-23T22:38:25Z</timestamp>
      <contributor>
        <username>Tajotep</username>
        <id>29695403</id>
      </contributor>
      <comment>/* Individualist anarchism */</
       comment>
      <model>wikitext</model>
      <format>text/x-wiki</format>
      <text xml:space="preserve">{{Use dmy dates|
       date=July 2018}}
{{redirect2|Anarchist|Anarchists|the fictional
character|Anarchist (comics)|other uses|Anarchists
(disambiguation)}}
{{pp-move-indef}}
{{use British English|date=January 2014}}
{{Anarchism sidebar
}}
{{Basic forms of government}}
. . .
. . .
      <sha1>9o00w06nsedf733vnmy7al780ia633d</sha1>
    </revision>
  </page>
```

Handling overly large files has been a major challenge of this project.

Each page within the format of the XML dump file has the following tags:

Title: contains the title of the article,
Id: the internal id of the article,
Redirect: what the page redirects to,
Namespace: helps identify the kind of page it is, and
Text: contains the information of the article itself.

HANDLING LARGE FILES

Handling overly large files has been a major challenge of this project. Regular Python parsing, which would require at least 70GB of available memory to load the XML file, is not feasible. Using the ElementTree package will allow us to load the XML file piece-by-piece without running into any memory issues. Being able to clear an element that was just processed frees memory space, allowing the next element to be loaded for processing.

```
import xml.etree.ElementTree as etree
[..]
for event, elem in etree.iterparse(pathWikiXML,
events=('start', 'end')):
    tname = strip_tag_name(elem.tag)
     ...
    elem.clear()
```

WIKIPEDIA TEMPLATES

Wikipedia uses Wiki Markup language to create all of its articles. The focus of each search is based on a specific medical condition. The following is a template that can be used to extract specific pieces of information including a condition's commonly prescribed medications, prognosis and mortality statistics:

```
{{Infobox medical condition (new)
| name           = <!--{{PAGENAME}} by default-->
 . . .
| medication     =
| prognosis      =
| frequency      =
| deaths         =
}}
```

Furthermore, the following template can explore a specific drug in relation to its brand and genericname as well as to its chemical properties:

```
{{Infobox drug
| drug_name         =
| INN               =
| type              =<!-- empty -->
| IUPAC_name        =
| image             =
| alt               =
| caption           =
<!-- Clinical data -->
| pronounce         =
. . .
| legal_status      = <!-- Free text -->
<!-- Pharmacokinetic data -->
| bioavailability   =
| protein_bound     =
. . .
| excretion         =
<!-- Identifiers -->
| CAS_number        =
. . .
| PubChem           =
| UNII              =
| DrugBank          =
<!-- Chemical and physical data -->
| chemical_formula  =
| molecular_weight  =
}}
```

MINING DRUG INFORMATION

The following template format enables the user to parse drug information data in a systematic way. Using string manipulation, for example, any page that contains the Drugbox template can be detected:

```
def get_drugbox(s):
    beg = (s.rfind('{{Drugbox'))
    end  =(s.rfind('\n}}'))
    if( end == -1):
        end = end =(s.rfind('}}\n'))
    if( end == -1):
        end = end =(s.rfind('}}\n<!--'))
    if( end == -1):
        end = end =(s.rfind('}}\n=='))
    if( end == -1):
        end = end =(s.rfind('}}\n\d'))
    s = s[beg: end+2]
```

After retrieving the Drugbox information, all of the data can be successfully mined. A range of codes can then be extracted from drug profiles such as CAS number, ATC and unique ingredient identifier (UNII) in addition to its chemical composition, most of which are unique to each English-speaking country.

For this project, the UNII code is of particular interest. The UNII is a unique, non-proprietary, free, unambiguous, non-semantic, alphanumeric identifier linked to a substance's molecular structure or descriptive information by the Substance Registration System (SRS) of the Food and Drug Administration (FDA) and the United States Pharmacopeia (USP). Below is the code required to extract the UNII code from the Drugbox template:

```
def find_unii(s):
    s = re.findall(r'UNII\s*?=\s?.*',s)
    if(len(s)>0):
        s = s[0]
        equal = s.rfind('=')
        #if there is a space after the equal
        remove it
        if(s[equal+1]==' '):
            s = s[equal+2:]
        else:
            s = s[equal+1:]
    else:
        s = ''
    return s
```

MINING SPECIFIC INFORMATION FOR EACH DISEASE

Due to the fact that the data in Wikipedia is typed manually by millions of individuals, various spacing and formats must be identified. By using the "InfoBox medical condition template" below, information about each disease can be extracted:

```
def get_med_cond(s):
    beg = (s.rfind('{{Infobox medical
    condition'))
    end  =(s.rfind('\n}}'))
    if( end == -1):
        end = end =(s.rfind('}}\n'))
    if( end == -1):
        end = end =(s.rfind('}}\n<!--'))
    if( end == -1):
        end = end =(s.rfind('}}\n=='))
    if( end == -1):
        end = end =(s.rfind('}}\n\d'))
    s = s[beg: end+2]
    return s
```

Specific information, such as symptoms, duration, causes, risks, medications and mortality, can be extracted from this template. Using the following code, the medications that are typically prescribed for a certain disease are able to be extracted:

```
def find_medication(s):
    s = re.findall(r'medication\s*?=.*<',s)
    if(len(s)== 0 ):
        s = []
    else:
        s = s[0]
        s = s[:-1]
        s = s[s.find('=')+2:]
        s = s.replace('[[','')
        s = s.replace(']]','')
        s = s.split(',')
        s
    return s
```

The World Health Organization (WHO) provides the medical classification codes through the International Statistical Classification of Diseases and Related Health Problems (ICD). Specifically, ICD9 and ICD10 codes can be extracted from the drug information present on Wikipedia. The following code shows how to obtain the medical codes for diseases, signs and symptoms, abnormal findings, complaints, social circumstances, and external causes of injury or diseases.

```
def find_icd10(s):
    s = s.replace('|','')
    s = s.replace('{{ICD10',"")
    icd10 = re.findall('\w{1}\d{2,6}',s)
    return icd10
```

CONCLUSION

After all of this data has been extracted it needs a place to be stored. Using a Coma Separated Values (CSV) file is a clean and easy way to store our data. Table 1 is an example on how the UNNI codes can be stored.

Table 1
Stored UNNI Codes

id	name	unii
1912	Ampicillin	7C782967RD
6346	Chloramphenicol	66974FR9Q1
10024	MDMA	KE1SEN21RM
11725	Flunitrazepam	620X0222FQ
14413	Hydrocodone	6YKS4Y3WQ7

Table 2
Stored Diseases, Codes and Meds

id	name	icd9	icd10	medications
4531	Bipolar disorder	['324.0']	['Q273,' 'q20,' 'Q280,' 'q20,' 'Q282,' 'q20']	['Lithium (medication)|Lithium,' ' antipsychotics', ' anticonvulsants']
4581	Bacterial vaginosis	['616.1']	['N76']	['Clindamycin or metronidazole']
4746	Plague (disease)	['020']	['A20']	['Gentamicin and a fluoroquinolone']
5876	Coronary artery disease	['780.0']	['R402,' 'r40']	['Aspirin,' ' beta blockers,' ' Medical use of nitroglycerin|nitroglycerin,' 'statins']

Furthermore, we can store the diseases with its codes and medications using arrays such that it is easier to then access the data when stored as a CSV file (See Table2).

Extrapolating data from Wikipedia can be challenging due to the amount of data it possesses and its inconsistencies. Nonetheless, Wikipedia provides diverse amounts of data that can be used by insurance companies to extract knowledge that otherwise would not be possible. The code provided in the article can be found in a Jupyter Notebook format on GitHub.[2] ■

Alexandru Loan Andrei is a Software Engineer at Reinsurance Group of America (RGA) in Chesterfield, Mo. He can be reached at *Alexandru.Andrei@rgare.com*

ENDNOTES

1 *https://dumps.wikimedia.org/enwiki/latest/*

2 *https://github.com/AlexAndrei98/*