Data and Text Processing for Health and Life Sciences

Francisco M. Couto

July 12, 2023

http://creativecommons.org/licenses/by/4.0/


Introduction

Health and Life studies known for the huge amount of data they produce.

Value of the data should not be measured by its amount, possibility and ability of researchers to retrieve and process it.

Transparency, openness, and reproducibility are key aspects.
Introduction

Biomedical data repositories

Examples: European Bioinformatics Institute (EBI)
          National Center for Biotechnology Information (NCBI) repositories

Researchers cannot rely on available data as mere facts
          they may contain errors
          can be outdated
          and may require a context
Structured data is what most computer applications require as input.

Humans tend to prefer the flexibility of text to express their hypothesis, ideas, opinions, conclusions.

Text is still the preferential means to publish new discoveries and to describe the data that support them.
Introduction

Amount of text

Huge amount of text being published every day

For example, 952,919 citations were added in 2020 to MEDLINE
10 articles per day
more than 261 years to read those articles

Scientific articles are not the only source of biomedical text
for example clinical studies and patents
Introduction

Total number of citations in MEDLINE

(Source: [https://www.nlm.nih.gov/bsd/](https://www.nlm.nih.gov/bsd/))
Total number of registered studies (clinical trials)

(Source: [https://clinicaltrials.gov](https://clinicaltrials.gov))
Introduction

Total number of patents in force

(Source: WIPO statistics database [http://www.wipo.int/ipstats/en/])
Ambiguity and contextualization

Inconsistency of the nomenclature
   Different terms referring to the same biological entity (synonyms)
   or the same term meaning different biological entities (homonyms)

The second problem is the complexity of the message
   Almost everyone can read and understand a newspaper story
   but just a few can really understand a scientific article

Finally the multilingual aspect of text is also a problem
   since most clinical data are produced in the native language
Biomedical ontologies

Address the issue of ambiguity of natural language and contextualization of the message

Vocabularies to guide what to look for
   official names and synonyms are used to mention concepts

Semantic models by providing semantic relationships between concepts
Programming skills

Data analysis is no longer possible using an in-house and limited dataset

How can I deal with such huge amount of data and text without having the necessary expertise, time and disposition to learn computer programming?
Why this book?

State-of-the-art tools based on complex and sophisticated technologies require knowledge on programming, linguistics, machine learning or deep learning impenetrable to the common Health and Life specialists and usually become outdated or even unavailable.

Set of skills to process text with minimal dependencies to existing tools and technologies.

Create a resilient and versatile solution with acceptable results.
Physician knows that the most efficient treatment for a specific patient is not always the most advanced one.

Data scientist knows that the most efficient tool to address a specific information need is not always the most advanced one.

Provides basic knowledge and right references to pursue a more advanced solution if required.
**Third-party solutions**

A common problem is their resiliency to deal with new user requirements and changes on how resources are being distributed. Software and hardware updates are crucial.

Commercial solutions tend to be more resilient. They need the funding to buy the service and are dependent on a third-party availability.

Open-source solutions may seem a great alternative, but they derive from academic projects that fade away to minimal updates. Using legacy software is a non-trivial task.
Simple pipelines

Build a simple pipeline with minimal software dependencies
not a fancy web front-end
manipulate our data using the spreadsheet application
automatize some of the repetitive tasks

In summary, this book is directed mainly towards Health and Life specialists and students that need to know how to process biomedical data and text, without being dependent on continuous financial support, third-party applications, or advanced computer skills
Shell scripting

Available for more than four decades
and now in almost every personal computer
Linux, macOS or Windows operating systems

But a shell script is still a computer algorithm, so how is it different from learning another programming language?
Usage of single command line tools
        combined as simple pipelines
        not intend to create experts in shell scripting

Demonstrate the ability of a few command line tools

Comprehending them is like conducting a new laboratory protocol
        testing and understanding its multiple procedural steps
        variables, and intermediate results
Text files

Data will be stored in text files that command line tools efficiently process. Simple and universal medium of storing data, opened and interpreted by using any text editor with a .txt extension without any kind of formatting.

Specific formats:

- CSV: Comma-Separated Values
- TSV: Tab-Separated Values
- XML: eXtensible Markup Language
Open (import), edited and saved (export) by any text editor application spreadsheet applications: such as LibreOffice Calc or Microsoft Excel

Spreadsheet example
CSV file contents:

A, C
G, T

TSV file contents:

A  C
G  T
**XML file contents:**

```xml
...<Table ss:StyleID="t1">
  <Column ss:Span="1" ss:Width="64.01"/>
  <Row ss:Height="12.81"><Cell><Data ss:Type="String">A</Data></Cell><Cell><Data ss:Type="String">C</Data></Cell></Row>
  <Row ss:Height="12.81"><Cell><Data ss:Type="String">G</Data></Cell><Cell><Data ss:Type="String">T</Data></Cell></Row>
</Table>
...```

---

*Data and Text Processing for Health and Life Sciences* by F. Couto

Slide 20 of 403
XLS file a lot of strange characters
   not a text file is a proprietary format

Comma-separated values is a data format so old as shell scripting
   in 1972 it was already supported by an IBM product

Using CSV or TSV enables us to manually manipulate the data
   using our favorite spreadsheet application
   and at the same time use command line tools
Relational databases

More advanced data storage techniques, still be able to use shell scripting import and export the data to CSV using sqlite3
What is in the book?

Chapter Resources
most prominent resources of biomedical data, text, and semantics
type of information they distribute
where we can find them
how we will be able to automatically explore them

Most of the examples use the resources provided by
the European Bioinformatics Institute (EBI)
using their services to automatically retrieve data

Not hard to adapt them to other service provider
such as the National Center for Biotechnology Information (NCBI).
Examples use two ontologies
one about human diseases
the other about chemical entities of biological interest.
share the same structure and syntax
Chapter Data Retrieval
   manual steps to retrieve text about \textit{caffeine}
   then automated by using command line tools

Step-by-step and introduce how each command line tool
can be used to automate each task
Command line tools

- **curl**: a tool to download data and text from the web;
- **grep**: a tool to search our data and text;
- **cut**: a tool to filter sections of each data item;
- **sed**: a tool to edit our data and text;
- **xargs**: a tool to repeat the same step for multiple data items;
- **xmlint**: a tool to search in XML data files.

- **cat**: a tool to get the content of file;
- **tr**: a tool to replace one character by another;
- **sort**: a tool to sort multiple lines;
- **head**: a tool to select only the first lines.
Pipelines

Redirect the output of a command line tool as input to another tool, or to a file sequential invocations of command line tools
Chapter Text Processing
extracting useful information from the text
finding references to *malignant hyperthermia*
*caffeine* related texts
Regular Expressions

Powerful pattern matching technique
- `grep` command line tool
- perform Named-Entity Recognition (NER)

Regular expressions originated in 1951
- even older than shell scripting
- still popular and available

String that include special operators
- represented by special characters

For example: `A | C | G | T`
Tokenization
identifying the text boundaries
such as splitting a text into sentences

Relation Extraction
find two entities in the same sentence,
Semantics

Ontologies

- construct large lexicons
- all entities of a domain (humans diseases)
- expand search using ancestors and related classes

Entity Linking

- each entity recognized is mapped to a ontology class
- deals with the ambiguity issue
- same label can be mapped to multiple classes
Resources

What is in the book?
Biomedical Text

Text preferential means publishing knowledge multiple types of sources main being scientific articles and patents less formal texts: electronic health records
What?

**Statement:** a short piece of text
- personal remarks
- evidence about a biomedical phenomenon

**Abstract:** a short summary of a larger scientific document

**Full-text:** the entire text present in a scientific document
Statements
  more syntactic and semantic errors
  not peer-reviewed
  directly linked to data
  brief and succinct form

Abstracts
  intellectual exercise to summarize
  may be insufficient to draw a solid conclusion
  details in a full-text document
Full-text documents
   may have restricted access.
structure and format varies
more information does not mean is beneficial
may even induce us in error
a fact in the Results vs. Related Work
Where?

PubMed
an information retrieval system
search and find biomedical texts
developed and maintained by NCBI

More than 28 million citations from MEDLINE
with title, abstract, authors, journal, publication date

EBI services
Europe PMC
Universal Protein Resource (UniProt) citations service
Alternative tools:
- Google Scholar
- Google Patents
- ResearchGate
- Mendeley
Tools also integrate semantic links
  - GOPubMed categorized texts using Gene Ontology
  - PubTator annotated with biological entities

Open Access Publications
  - full-texts freely available with unrestricted use
  - PubMed Central (PMC) more than 5 million documents
Electronic health records
  stored in health institutions and linked to patients
  access is restricted due to ethical and privacy issues

THYME corpus
  more than one thousand de-identified clinical notes (Mayo Clinic)
  only available for text processing research

Social networks
  identify new trends and insights about a disease
  processing tweets to predict flu outbreaks
Programmatic access to it
restrictions only manual access is granted
face a CAPTCHA challenge (humans or not)

NCBI and EBI online services
such as PubMed, Europe PMC, or UniProt Citations
allow programmatic access with Web APIs

RESTful web services (simple uniform interface)
Uniform Resource Locator (URL) self-explanatory
enough to retrieve the data using command line
Lack of use of standard nomenclatures
- different labels (synonyms, acronyms)
- sharing the same label (homonyms)
  requires sense disambiguation to select the correct meaning

Disease acronym ATS may represent
- *Andersen-Tawil syndrome*
- or the *X-linked Alport syndrome*

Solution: ontologies and semantic similarity
What?

In 1993 definition of ontology:

an explicit specification of a conceptualization

In 1997 and 1998 refined to:

a formal, explicit specification of a shared conceptualization
Conceptualization
  an abstract view of the concepts
  and the relationships of a given domain

Shared conceptualization
  a group of individuals agree (common agreement)

Specification is a representation of that conceptualization
  using a given language.
  needs to be formal and explicit
  so computers can deal with it
Languages

Web Ontology Language (OWL)
   most common languages to specify ontologies

Open Biomedical Ontology (OBO)
   principles to ensure high quality, formal rigor
   and interoperability between other OBO ontologies
Concepts are defined as OWL classes that may include multiple properties, such as labels official name, acronyms, exact synonyms, and even related terms

Class *malignant hyperthermia*

   synonym *anesthesia related hyperthermia*.

*Andersen-Tawil syndrome* and *X-linked Alport syndrome*

   share *ATS* as an exact synonym
Formality

Different levels of formality such as controlled vocabularies, taxonomies and thesaurus may include logical axioms.

Controlled vocabularies are list of terms without specifying any relation between them.

Taxonomies are controlled vocabularies that include subsumption relations.

*malignant hyperthermia* is a *muscle tissue disease*
is-a or subclass relations
are normally the backbone of ontologies.
include multiple inheritance
organized as a directed acyclic graphs (DAG)

Thesaurus
includes other types of relations besides subsumption
caffeine has role mutagen.
DAG representing a classification of metals with multiple inheritance
Gold related documents

Find texts related to *gold*

a corpus with one distinct document mentioning each metal except for *gold* that no document mentions

which documents should we read first?

*silver* is probably the most related

shares two parents, *precious* and *coinage*. 
platinum, palladium or copper?
  depends on our information need
  previous searches or reads

Last searches were coinage
  copper is probably the second-most related

Importance of these semantic resources
  development of the knowledge graph by Google
BioPortal (April 2021)
  858 ontologies
  more than 10 million classes

Search for *caffeine*
  large list of ontologies that define it
  conceptualizations of *caffeine* in different domains
  alternative perspectives
  Interoperability property with links to similar classes

OBO initiative
  tackle this somehow disorderly spread of definitions
  each OBO ontology covers a clearly specified scope
Resources

OBO ontologies

Success of Gene Ontology (GO)
  describe molecular function, biological process and cellular component
gene-products for different species

Disease Ontology (DO)
  human disease terms
  phenotype characteristics
  and related medical vocabulary disease concepts

Chemical Entities of Biological Interest (ChEBI)
  classification of molecular entities
  with biological interest
  focus on small chemical compounds
Popular controlled vocabularies

International Classification of Diseases (ICD)
  by World Health Organization (WHO)
  generic clinical terms

Systematized Nomenclature of Medicine - Clinical Terms (SNOMED CT)
  highly comprehensive and detailed

Medical Subject Headings (MeSH)
  classifying biomedical and health-related information and documents

Unified Medical Language System (UMLS)
  large resource integrate most biomedical vocabularies
  2015AB release more than three million concepts
Ontobee
  repository of ontologies (most OBO ontologies)
  231 ontologies (April 2021)

Outside the biomedical domain
  W3C SWEO Linking Open Data community project
  W3C Library Linked Data Incubator Group
How?

Find ontology home page
  download the most recent release
  the original format
  select the subset of the ontology

ChEBI provides three versions:
  LITE, CORE and FULL

If not interested in chemical data and structures
  that is available in CORE
  LITE is probably the best solution
  may miss synonyms from FULL version
OWL language prevailing language to represent ontologies

OWL extends RDF Schema (RDFS)
with more complex statements using description logic

RDFS is an extension of RDF
with additional statements
such as class-subclass or property-subproperty relationships
RDF is a data model
stores information in statements
represented as triples: subject, predicate and object

RDF data encoded using Extensible Markup Language (XML)
named RDF/XML

XML is a self-descriptive mark-up language
composed of data elements
XML example

*caffeine* is a drug
may treat the condition of sleepiness
without being an official treatment:

```xml
<treatment category="non-official">
  <drug>caffeine</drug>
  <condition>sleepiness</condition>
</treatment>
```

Hierarchical structure of data elements:

- new data element
- data element will end

*property* `category` **with value** `non-official`
Large XML files are almost unreadable by humans

N3 and Turtle
  legible encoding languages for RDF

Most biomedical ontologies in OWL using XML encoding
The Uniform Resource Identifier (URI) is a standard global identifier of classes. The class *caffeine* in ChEBI is identified by the URI:

http://purl.obolibrary.org/obo/CHEBI_27732

A URI is a URL if we open it in a web browser and obtain a resource describing that class.
Ontologies are also available as database dumps.
  normally SQL files
  in a DataBase Management System (DBMS)

Use command line tool **sqlite3**
  execute the SQL commands
  import data (.read command)
  export data to CSV (.mode command)
Data Retrieval

Example of how we can retrieve text manually and then automatize using shell script commands
Caffeine Example

Retrieve data and literature about caffeine searching caffeine in Wikipedia information available in the infobox multiple links to external sources

Link to resource hosted by the European Bioinformatics Institute (EBI) CHEBI:27732 - Chemical Entities of Biological Interest includes an ontology with structural and biological properties
Caffeine is a central nervous system (CNS) stimulant of the methylxanthine class. It is the world's most widely consumed psychoactive drug. Unlike many other psychoactive substances, it is legal and unregulated in nearly all parts of the world. There are several known mechanisms of action to explain the effects of caffeine. The most prominent is that it reversibly blocks the action of adenosine on its receptor and consequently prevents the onset of drowsiness induced by adenosine. Caffeine also stimulates certain portions of the autonomic nervous system.

Caffeine is a bitter, white crystalline purine, a methylxanthine alkaloid, and is chemically related to the adenine and guanine bases of deoxyribonucleic acid (DNA) and ribonucleic acid (RNA). It is found in the seeds, nuts, or leaves of a number of plants native to Africa, East Asia
Identifiers section of the Wikipedia page about caffeine
ChEBI entry describing caffeine
External references related to caffeine

**Protein Sequences**

- **UniProt KB**
  - *UniProt Knowledge Base of protein sequences.*
  - **A2AGL3**
    - Ryanodine receptor 3
  - **A4GE99**
    - 7-methylxanthosine synthase 1
  - **A4GE70**
    - 3,7-dimethylxanthine N-methyltransferase
  - **A6MFK9**
    - Cysteine-rich venom protein
  - **BOLPN4**
    - Ryanodine receptor 2

**Small molecules**

- **NMRShiftDB**
  - *NMRShiftDB is a NMR database for organic structures and their nuclear magnetic resonance (nmr) spectra.*
  - **10016316**

**Reactions & Pathways**

- **BioModels**
  - *Database of Mathematical models of biological interest.*
  - **BIOMD0000000241**
    - Shi93_Caffeine_pressor_tolerance
  - **BIOMD0000000601**
    - Rosas2015 - Caffeine-induced luminal SR calcium changes

- **BKMS-react**
  - *BKMS-react is an integrated and non-redundant biochemical reaction database containing known enzyme-catalyzed and spontaneous reactions.*
  - **882**
  - **7965**
  - **51266**

- **Rhea**
  - *Rhea is a freely available, manually annotated database of biochemical reactions.*

---

*Data and Text Processing for Health and Life Sciences* by F. Couto  
Slide 68 of 403
### UniProt Automatically Generated Cross-References

Version 2014_02 of UniProt was used for these cross-references.

77 entries found, displaying 1 to 15.

<table>
<thead>
<tr>
<th>Identifiers</th>
<th>Name</th>
<th>Line Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>A2AGL3</td>
<td>Ryanodine receptor 3</td>
<td>CC - MISCELLANEOUS</td>
</tr>
<tr>
<td>A4GE69</td>
<td>7-methylxanthosine synthase 1</td>
<td>CC - FUNCTION</td>
</tr>
<tr>
<td>A4GE70</td>
<td>3,7-dimethylxanthine N-methyltransferase</td>
<td>CC - CATALYTIC ACTIVITY; CC - FUNCTION</td>
</tr>
<tr>
<td>A6MFK9</td>
<td>Cysteine-rich venom protein</td>
<td>CC - FUNCTION</td>
</tr>
<tr>
<td>B0LPN4</td>
<td>Ryanodine receptor 2</td>
<td>CC - MISCELLANEOUS</td>
</tr>
<tr>
<td>B7FDI0</td>
<td>Cysteine-rich venom protein</td>
<td>CC - FUNCTION</td>
</tr>
<tr>
<td>B7FDI1</td>
<td>Cysteine-rich venom protein</td>
<td>CC - FUNCTION</td>
</tr>
<tr>
<td>B8QG90</td>
<td>Hadruclacin</td>
<td>CC - FUNCTION</td>
</tr>
<tr>
<td>D7REY3</td>
<td>Caffeine dehydrogenase subunit alpha</td>
<td>DE; FT; CC - CATALYTIC ACTIVITY; CC - FUNCTION; CC - BIOPHYSICOCHEMICAL PROPERTIES</td>
</tr>
<tr>
<td>D7REY4</td>
<td>Caffeine dehydrogenase subunit beta</td>
<td>DE; FT; CC - CATALYTIC ACTIVITY; CC - FUNCTION; CC - BIOPHYSICOCHEMICAL PROPERTIES</td>
</tr>
<tr>
<td>D7REY5</td>
<td>Caffeine dehydrogenase subunit gamma</td>
<td>DE; FT; CC - CATALYTIC ACTIVITY; CC - FUNCTION; CC - BIOPHYSICOCHEMICAL PROPERTIES</td>
</tr>
<tr>
<td>E9PZQ0</td>
<td>Ryanodine receptor 1</td>
<td>CC - MISCELLANEOUS</td>
</tr>
<tr>
<td>E9Q4O1</td>
<td>Ryanodine receptor 2</td>
<td>CC - MISCELLANEOUS</td>
</tr>
<tr>
<td>F0E1K6</td>
<td>Probable methylxanthine N7-demethylyase NdNC</td>
<td>CC - FUNCTION</td>
</tr>
<tr>
<td>F1LMY4</td>
<td>Ryanodine receptor 1</td>
<td>CC - MISCELLANEOUS</td>
</tr>
</tbody>
</table>

Proteins related to caffeine
Protein Sequences section
Click on *show all* to complete list
includes the identifiers of each protein

UniProt
a database of protein sequences and annotation data.
resource hosted EBI

DISRUPTION PHENOTYPE
effects caused by the disruption of the gene coding
Bottom-right *Export options*
CSV, Excel and XML files

Open in a text editor software
notepad (Windows), TextEdit (macOS) or gedit (Linux).
chebi_27732_xrefs_UniProt.csv:
- A2AGL3, Ryanodine receptor 3, CC - MISCELLANEOUS
- A4GE69, 7-methylxanthosine synthase 1, CC - FUNCTION
...

chebi_27732_xrefs_UniProt.xls:

<table>
<thead>
<tr>
<th>Identifiers</th>
<th>Name</th>
<th>Line Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>A2AGL3</td>
<td>Ryanodine receptor 3</td>
<td>CC - MISCELLANEOUS</td>
</tr>
<tr>
<td>A4GE69</td>
<td>7-methylxanthosine synthase 1</td>
<td>CC - FUNCTION</td>
</tr>
</tbody>
</table>
...

chebi_27732_xrefs_UniProt.xml:

```xml
<?xml version="1.0"?>
<table>
  <row>
    <column>A2AGL3</column>
    <column>Ryanodine receptor 3</column>
    <column>CC - MISCELLANEOUS</column>
  </row>
  <row>
    <column>A4GE69</column>
    <column>7-methylxanthosine synthase 1</column>
    <column>CC - FUNCTION</column>
  </row>
  ...
```
UniProt entry describing the Ryanodine receptor 1
Select *Ryanodine receptor 1* P21817
more than just sequence database

Click on Format and on XML.
save the result as a XML file
**P21817.xml:**

```
<?xml version='1.0' encoding='UTF-8'?>
<uniprot xmlns="http://uniprot.org/uniprot" ...>
<entry dataset="Swiss-Prot" created="1991-05-01" ...>
<accession>P21817</accession>
...
```
*Homo sapiens* (*Human*) protein

interested only in Human Proteins

filter them

Entry E9PZQ0

*Ryanodine receptor 1* protein

but *Mus musculus* (*Mouse*)
Publications related to Ryanodine receptor 1
Top-left link to publications
   click on it
   list of publications somehow related to the protein
Abstract of the publication entitled *Polymorphisms and deduced amino acid substitutions in the coding sequence of the ryanodine receptor (RYR1) gene in individuals with malignant hyperthermia*
Finding phenotypic information, the first title that may attract our attention:

*Polymorphisms and deduced amino acid substitutions in the coding sequence of the ryanodine receptor (RYR1) gene in individuals with malignant hyperthermia*

Clicking on the Abstract link
Diseases recognized by the online tool MER in an abstract
Mentions any disease
   use an online text mining tool
   Minimal Named-Entity Recognizer (MER)
   copy and paste the abstract
   select DO - Human Disease Ontology as lexicon

Detects three mentions of malignant hyperthermia,
   link about the disease
Disease Ontology - Institute

Search Ontology...

Welcome

DOID:8545

Metadata

ID

DOID:8545

Name

malignant hyperthermia

Definition

A muscle tissue disease that is characterized by a drastic and uncontrolled increase in skeletal muscle oxidative metabolism, which overwhelms the body's capacity to supply oxygen, remove carbon dioxide, and regulate body temperature and that has material basis in heterozygous mutation in the ryanodine receptor gene (RyR1) on chromosome 19q13. Manifestations of malignant hyperthermia (MH) are precipitated by certain volatile anesthetics (i.e., halothane, isoflurane, sevoflurane, desflurane, enfurane), either alone or in conjunction with a depolarizing muscle relaxant (specifically, succinylcholine).

http://en.wikipedia.org/wiki/Malignant_hyperthermia

Synonyms

GARD:6964
ICD10CM.T68.3
ICD9CM.565.86
MESH:D008305
NCI:C34369
Owls.B6546C
ORDO:423
SNOMEDCT_US_2021_09_01:111738008
UMLS_CUI:C0024591

Cro...
Entry for the class *malignant hyperthermia*
Need to repeat all the steps to all the proteins
   all publications of each protein

More complicated if all central nervous system stimulants

Motivation to automatize the process,
   not humanly feasible
Goal relation between *caffeine* and hyperthermia, simply search these two terms in PubMed

1 - Some relations are not explicitly mention in the text

2 - Example using different resources and multiple entries to automate using shell scripting
Unix shell

A shell is a software program
  interprets and executes command lines
  given by the user in consecutive lines of text

A shell script is a list of such command lines

The command line usually starts by invoking a command line tool.
Unix shell was developed to manage Unix-like operating systems, nowadays available is most computers Linux, macOS or Windows operating systems.

Types of Unix shells with minor differences between them sh, ksh, csh, and tcs most widely available is the Bourne-Again shell (bash)
Debian GNU/Linux 6.0 debian1 tty1

debian1 login: root
Password:
Last login: Fri Jan 10 09:53:12 CST 2014 on tty1
Linux debian1 2.6.32-5-686 #1 SMP Fri May 10 08:33:48 UTC 2013 i686

The programs included with the Debian GNU/Linux system are free software; the exact distribution terms for each program are described in the individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.
root@debian1:~# _

Screenshot of a Terminal application
Linux or macOS
  terminal application already installed
  that opens a shell for us.

Microsoft Windows operating system
  Windows 10 install a Windows Subsystem for Linux
  or a third-party application, such as MobaXterm
Current directory

First command line:

```
$ pwd
```

Shows the full path of the directory (folder) in which the shell is working on.
The dollar sign in the left
  a command to be executed directly in the shell

A curved arrow in the right
  a command does not fit in the available width of a page
  and has to be presented in multiple lines
To understand a command line tool
type `man` followed by the name of the tool.
For example `man pwd`

Or type `pwd --help`

a more concise description of `pwd`. 
ls
  shows list of files in the current directory.

Type `ls --help`
  a concise description of ls

Select a current directory
  we can easily open in our file explorer application
Windows directories

Separated by a backslash (\)
in a Unix shell is a forward slash (/).

Windows path to the Documents folder:
C:\Users\MyUserName\Documents

Windows Subsystem for Linux:
/mnt/c/Users/MyUserName/Documents

MobaXterm:
/drives/c/Users/MyUserName/Documents
Change directory

Type `cd` (change directory) followed by the new path:

```
$ cd Documents
```

Type `pwd` to see what changed.
Return to the parent directory:
$ cd ..

Return to the home directory:
$ cd ~

Windows full path:
$ cd /mnt/c/Users/MyUserName/Documents
Enclose the path within single (or double) quotes in case it contains spaces:

```
$ cd '/mnt/c/Users/MyUserName/Documents'
```

Later on difference between using single or double quotes. we may assume that they are equivalent.
Useful key combinations

Terminal is blocked
press Ctrl-c cancels the current tool

For example: try using the cd command with only one single quote:
```
$ cd '
```

Now press Ctrl-c, and the command will be aborted.

Ctrl-d indicates the terminal that it is the end of input. command will not be canceled, executed without the second single quote a syntax error will be shown on our display
Ctrl-l cleans the terminal display

control-insert and shift-insert

copy and paste the selected text
Shell version

Check if the output says bash:

$ ps -p $$

ps shows information about active processes

The -p option selects a given process,

$$ represents the process running in our terminal
Data file

Create a file named *myfile.txt* using any text editor:

```
line 1
line 2
line 3
line 4
```

Save it in working directory

check proper filename extension.
File contents

Type:

```
$ cat myfile.txt
```

cat receives a filename as argument
displays its contents on the screen.
Reverse file contents

Type:
$ tac myfile.txt

The contents of the file in the reverse order

In macOS use tail -r
My first script

Create a script file named `reversemyfile.sh`:

```
1 tac $1
```

$1 represents the first argument

Each script will include the line numbers in the left
helps identify how many lines
and distinguish from commands directly in the shell
Line breaks

A Unix file a line break is a line feed character instead of two characters (carriage return and line feed) used by Windows.

Text editor in Windows save it as Unix file, open source Notepad++

Text editor in macOS save it in text format
Remove the extra carriage return:

$ tr -d '\r' < reversemyfile.sh > reversemyfilenew.sh

The \texttt{-d} option of \texttt{tr} removes a given character from the input.

This case delete all carriage returns (\texttt{r})

Command line options can be used in short form using a single dash (\texttt{-}) or in a long form using two dashes (\texttt{--})

\texttt{--delete} is equivalent to \texttt{-d}
Redirection operator

> character
  moves the results being displayed at the standard output (our terminal) to a given file.

< character
  works on the opposite direction
  opens a given file
  uses it as the standard input
cat filename as an input argument
while tr through the standard input

cat can also receive contents through the standard input:

$ cat < myfile.txt
`tr` a new file for the standard output
cannot use the same file to read and write

To keep the same filename use `mv`:

```
$ mv reversemyfilenew.sh reversemyfile.sh
```
Installing tools

Two last two commands replaced by dos2unix:

$ dos2unix -n reversemyfile.sh

If not available, install the dos2unix tool:

$ apt install dos2unix

In macOS:

$ brew install dos2unix
Avoid fixing line breaks each time update file using Windows, better solution is a Unix friendly text editor

Using a Unix friendly text editor, the previous commands nothing will happen since `tr` not remove any character
Permissions

A script also needs permission to be executed:

```
$ chmod u+x reversemyfile.sh
```

The `chmod` command just gave the user (u) permissions to execute (+x)
Finally, execute the script:

```bash
$ ./reversemyfile.sh myfile.txt
```

Result:

- line 4
- line 3
- line 2
- line 1
More arguments will be ignored:
$ ./reversemyfile.sh myfile.txt myotherfile.txt 'my other file.txt'

The output will be exactly the same
does not use $2 and $3

When containing spaces
the argument enclosed by single quotes
Debug

Not working well debug the entire script:

```bash
$ bash -x reversemyfile.sh myfile.txt
```

Command line tools executed preceded by +:

```bash
+ tac myfile.txt
line 4
line 3
line 2
line 1
```

Or add `set -x` in script to start debug

and `set +x` to stop debug
Save output

Save output into another file:

$ ./reversemyfile.sh myfile.txt > mynewfile.txt

Check if the file was really created:

$ cat mynewfile.txt

Or reverse it again:

$ ./reversemyfile.sh mynewfile.txt
Web Identifiers

Input argument(s) of our task
- is the chemical compound(s)
- ChEBI identifier(s)
finding the identifier by its name is also possible

Retrieve all proteins
- associated to caffeine (CHEBI:27732).
Links shown as *Export options*:

https://www.ebi.ac.uk/chebi/viewDbAutoXrefs.do?d-1169080-e=1&6578706f7274=1&chebiId=27732&dbName=UniProt
https://www.ebi.ac.uk/chebi/viewDbAutoXrefs.do?d-1169080-e=2&6578706f7274=1&chebiId=27732&dbName=UniProt
https://www.ebi.ac.uk/chebi/viewDbAutoXrefs.do?d-1169080-e=3&6578706f7274=1&chebiId=27732&dbName=UniProt

Downloading a CSV, Excel, or XML file, respectively.

The only difference

single numerical digit (1, 2, and 3)
after the first =
argument to select the type of file
Another parameter
the ChEBI identifier (27732).

Replace 27732 by 17245 in any of those URLs:

https://www.ebi.ac.uk/chebi/viewDbAutoXrefs.do?d-1169080-e
=1&6578706f7274=1&chebiId=17245&dbName=UniProt

Downloads more than seven hundred proteins
since 17245 the ChEBI identifier
of carbon monoxide.
Not using a fully RESTful web service  
    but pretty modular and self-explanatory

The path is clearly composed of:

- the name of the database (chebi);
- the method (viewDbAutoXrefs.do);
- list of parameters and their value (arguments) after ?
Order of the parameters is normally not relevant separated by &
= assigns a value to each parameter (argument).

As data pipelines to fill our local files with data like pipelines that transport oil or gas
Script `getproteins.sh`:

1. `echo 'The input: $1'`
2. `echo "The input: $1"`

$ ./getproteins.sh

The input: $1
The input:

With an argument:

$ ./getproteins.sh 27732

The input: $1
The input: 27732
Comments

1  #echo 'The input: $1'
2  #echo "The input: $1"
3  echo "https://www.ebi.ac.uk/chebi/viewDbAutoXrefs.do?d-1169080-e=1&6578706f7274=1&chebiId=$1&dbName=UniProt"

Commented lines are ignored

ChEBI identifier as argument:

$ ./getproteins.sh 27732

Output the link for CSV file with proteins associated with caffeine
Client Uniform Resource Locator (cURL)
   a command line tool
download a URL directly into a file

List of proteins related to **caffeine**:

```bash
$ curl 'https://www.ebi.ac.uk/chebi/viewDbAutoXrefs.do?d-1169080-e♂
   =1&6578706f7274=1&chebiId=27732&dbName=UniProt'
```

...  
Q15413,Ryanodine receptor 3,CC - MISCELLANEOUS  
Q92375,Thioredoxin reductase,DE  
Q92736,Ryanodine receptor 2,CC - MISCELLANEOUS
Alternative:

$ wget -O- 'https://www.ebi.ac.uk/chebi/viewDbAutoXrefs.do?d-1169080-e=1&6578706f7274=1&chebiId=27732&dbName=UniProt'

Instead of using a fixed URL, update `getproteins.sh`:

```bash
1 curl "https://www.ebi.ac.uk/chebi/viewDbAutoXrefs.do?d-1169080-e=1&6578706f7274=1&chebiId=$1&dbName=UniProt"
```

Using double quotes
Execute:

```
$ ./getproteins.sh 27732
...
Q15413, Ryanodine receptor 3, CC - MISCELLANEOUS
Q92375, Thioredoxin reductase, DE
Q92736, Ryanodine receptor 2, CC - MISCELLANEOUS
```
Proteins related to *carbon monoxide*:

```
$ ./getproteins.sh 17245
...
Q58432,Phosphomethylpyrimidine synthase,CC - CATALYTIC ACTIVITY
Q62976,Calcium-activated potassium channel subunit alpha-1,CC - ENZYME REGULATION; CC - DOMAIN
Q63185,Eukaryotic translation initiation factor 2-alpha kinase 1,CC - ENZYME REGULATION
```
Command line tool `less`

navigate using the arrow keys

Bar character (\|) between two commands

transfer the output of the first command as input of the second

$ ./getproteins.sh 27732 | less

To exit press `q`.

Save the output as a file:

$ ./getproteins.sh 27732 > chebi_27732_xrefs_UniProt.csv

Download progress information still displayed
Standard error output

Redirect the standard error output (2) to the null device:

```bash
$ ./getproteins.sh 27732 > chebi_27732_xrefs_UniProt.csv 2>/dev/null
```

Or use `-s` option of `curl`:

```bash
1 curl -s "https://www.ebi.ac.uk/chebi/viewDbAutoXrefs.do?d=1169080-e=16578706f7274=1&chebiId=$1&dbName=UniProt"
```

The equivalent long form to the `-s` option is `--silent`. 
Check if created:

$ less chebi_27732_xrefs_UniProt.csv

Or use spreadsheet application
such as LibreOffice Calc or Microsoft Excel

Exercise get the CSV file
associated proteins of water and gold
Data Extraction

Select the relevant proteins (lines)

using `grep`,

Select the column

using `cut`
Diseases related to *caffeine*,
only interested in proteins (lines) with topics:
CC  -  MISCELLANEOUS  
CC  -  DISRUPTION PHENOTYPE  
CC  -  DISEASE

Extracting lines from a text file
main function of `grep`
diving as input a pattern
Single and multiple patterns

**Selects proteins topic** CC - MISCELLANEOUS:

```bash
$ grep 'CC - MISCELLANEOUS' chebi_27732_xrefs_UniProt.csv

A2AGL3, Ryanodine receptor 3, CC - MISCELLANEOUS
B0LPN4, Ryanodine receptor 2, CC - MISCELLANEOUS
...
Q15413, Ryanodine receptor 3, CC - MISCELLANEOUS
Q92736, Ryanodine receptor 2, CC - MISCELLANEOUS
```
Multiple patterns
precede with `-e` option:

```
$ grep -e 'CC - MISCELLANEOUS' -e 'CC - DISRUPTION PHENOTYPE' -e 'CC - DISEASE' chebi_27732_xrefs_UniProt.csv

...  
Q9VSH2,Gustatory receptor for bitter taste 66a,CC - FUNCTION; CC - DISRUPTION PHENOTYPE
Q15413,Ryanodine receptor 3,CC - MISCELLANEOUS
Q92736,Ryanodine receptor 2,CC - MISCELLANEOUS
```
Add | less to check carefully

less also find lines based on a pattern
  type /
  and then a pattern
Update *getproteins.sh*:

```bash
1 curl -s "https://www.ebi.ac.uk/chebi/viewDbAutoXrefs.do?d-
   -1169080-e=1&6578706f7274=1&chebiId=$1&dbName=UniProt" | \ 
2 grep -e 'CC - MISCELLANEOUS' -e 'CC - DISRUPTION PHENOTYPE' -e '˜
   CC - DISEASE'
```

Added `-s` to suppress the progress information

The characters | \ 
redirect the output of that line 
as input of the next line 
ensure \ is the last character
We can now execute the script again:

```
$ ./getproteins.sh 27732
```

The output should be similar of what we got previously, but the script downloads the data and filters immediately.

To save the file with the relevant proteins, we only need to add the redirection operator:

```
$ ./getproteins.sh 27732 > chebi_27732_xrefs_UniProt_relevant.csv
```
Data elements selection

Select first column
  the one that contains the protein identifiers
  one easy task for cut

Select character that divides each data element (column)
  with -F option,
  and instruction of what to do with it
  enclosed by single quotes and curly brackets
Get the first column of CSV file:

```
$ cut -d, -f1 < chebi_27732_xrefs_UniProt_relevant.csv

... 
Q9VSH2
Q15413
Q92736
```

Comma (,) character that separates data elements

- `print` is equivalent to `echo`
- `1` the first data element
Example for first and third columns:

$ cut -d, -f1,3 < chebi_27732_xrefs_UniProt_relevant.csv

... 
Q9VSH2, CC - FUNCTION; CC - DISRUPTION PHENOTYPE
Q15413, CC - MISCELLANEOUS
Q92736, CC - MISCELLANEOUS
Update `getproteins.sh`:

```bash
1 curl -s "https://www.ebi.ac.uk/chebi/viewDbAutoXrefs.do?d-
   -1169080-e=1&6578706f7274=1&chebiId=$1&dbName=UniProt" | \ 
2 grep -e 'CC - MISCELLANEOUS' -e 'CC - DISRUPTION PHENOTYPE' -e 'CC - DISEASE' | \ 
3 cut -d, -f1
```

The last line is the only that changes

except the | \
Execute:

$ ./getproteins.sh 27732

Output similar of what we got previously
but now only the protein identifiers

To save the output as a file:

$ ./getproteins.sh 27732 >
  chebi_27732_xrefs_UniProt_relevant_identifiers.csv
Given a protein identifier
construct the URL
download its information from UniProt
RESTful web services provided by UniProt

URL starts by https://www.uniprot.org/uniprot/
followed by the protein identifier
ending with a dot and the data format
https://rest.uniprot.org/uniprotkb/P21817.xml
Assembly line

Construct one URL for each protein from the previously list. The size can be large (hundreds of proteins) and varies for different compounds and evolves with time.

We need an assembly line: list of proteins identifiers added as input, construct one URL per protein and retrieve the respective file.

`xargs` works as an assembly line, executes a command per each line given as input.
Display each identifier:

```bash
$ cat chebi_27732_xrefs_UniProt_relevant_identifiers.csv | xargs -I {} echo 'Another protein id {} to retrieve'
```

...  
Another protein id Q9VSH2 to retrieve
Another protein id Q15413 to retrieve
Another protein id Q92736 to retrieve

Input the contents our CSV file
for each line displayed a message
-I replaces {} by the value of the line being processed
Create the URLs:

$ cat chebi_27732_xrefs_UniProt_relevant_identifiers.csv | xargs -I{} echo 'https://rest.uniprot.org/uniprotkb/{}.xml'

...  
https://rest.uniprot.org/uniprotkb/Q9VSH2.xml  
https://rest.uniprot.org/uniprotkb/Q15413.xml  
https://rest.uniprot.org/uniprotkb/Q92736.xml

Try these links in our internet browser
Download using the `curl`:

```bash
$ cat chebi_27732_xrefs_UniProt_relevant_identifiers.csv | xargs -I{} curl 'https://rest.uniprot.org/uniprotkb/{}.xml' -o 'chebi_27732_{}.xml'
```

- `-o` save the output to a given file named after each protein identifier
Check using `ls`:

```
$ ls chebi_27732_* .xml
```

* represents any file whose name starts with `chebi_27732_` and ends with `.xml`

Check contents:

```
$ less chebi_27732_P21817.xml
```
File header

Content has to start with `<?xml`
otherwise was download error
run `curl` again for those entries

Check the header with `head`:

```
$ head -n 1 chebi_27732_*.xml | less
```

`-n` specifies how many lines to print

Not able to download from UniProt
Variable

Update `getproteins.sh`:

```
1  ID=$1 # The CHEBI identifier given as input is renamed to ID
2  rm -f chebi\_${ID}\_*.xml # Removes any previous files
3  curl -s "https://www.ebi.ac.uk/chebi/viewDbAutoXrefs.do?d
   -1169080-e=1&6578706f7274=1&chebiId=$ID&dbName=UniProt" | \\
4  grep -e 'CC - MISCELLANEOUS' -e 'CC - DISRUPTION PHENOTYPE' -e 'CC - DISEASE' | \\
5  cut -d, -f1 | xargs -I {} curl 'https://rest.uniprot.org/
   uniprotkb/{}.xml' -o chebi\_${ID}\_{}.xml
```
Includes `xargs`, `curl` and `$ID` variable

variable contains value of `$1`

Preceding character of `$ID` an underscore (_)

add a backslash (`\`) before it

`rm` remove any files downloaded previously
Execute:

$ ./getproteins.sh 27732

Check results:

$ head -n 1 chebi_27732_*.xml | less
XML Processing

Only human diseases process the XML of each protein check if a *Homo sapiens* (*Human*) protein.
Use grep:

```
$ grep '<name type="scientific">Homo sapiens</name>' chebi_27732_*.xml
```

```
chebi_27732_P21817.xml:<name type="scientific">Homo sapiens</name>
chebi_27732_Q15413.xml:<name type="scientific">Homo sapiens</name>
chebi_27732_Q8N490.xml:<name type="scientific">Homo sapiens</name>
chebi_27732_Q92736.xml:<name type="scientific">Homo sapiens</name>
```
-l option just filename:

```bash
$ grep -l '<name type="scientific">Homo sapiens</name>'
  chebi_27732_*.xml
```

The output will now show only the filenames:

- `chebi_27732_P21817.xml`
- `chebi_27732_Q15413.xml`
- `chebi_27732_Q8N490.xml`
- `chebi_27732_Q92736.xml`
PubMed identifiers

Extract PubMed identifiers:

```bash
$ grep 'dbReference type="PubMed"' chebi_27732_P21817.xml  
...
<dbReference type="PubMed" id="11741831"/>
<dbReference type="PubMed" id="16163667"/>
<dbReference type="PubMed" id="27586648"/>
```
Just the identifier:

```bash
$ grep '<dbReference type="PubMed"' chebi_27732_P21817.xml | cut -d" -f4

... 
11741831
16163667
27586648
```

" as separation character
Pubmed identifiers extraction

Apply to every protein:

```bash
$ grep -l '<name type="scientific">Homo sapiens</name>' chebi_27732_*.xml | xargs -I {} grep '<dbReference type="PubMed" ' {} | cut -d" -f4
```

Long list of Pubmed identifiers
including repetitions
same publication cited in different entries
Duplicate removal

Identify the repetitions with `sort` repeated identifiers in consecutive lines

```
$ grep -l '<name type="scientific">Homo sapiens</name>' chebi_27732_*.xml | xargs -I {} grep '<dbReference type="PubMed"' {} | cut -d" -f4 | sort

...  
9607712  
9607712  
9607712
```
--u option removes duplicates:

```
$ grep -l '<name type="scientific">Homo sapiens</name>'
  chebi_27732_*.xml | xargs -I {} grep '<dbReference type="PubMed
  "' {} | cut -d" -f4 | sort -u
```

Check how many duplicates were removed

word count `wc` command

with and without the usage of --u:
$ grep -l '<name type="scientific">Homo sapiens</name>' chebi_27732_*.xml | xargs -I {} grep '<dbReference type="PubMed"' {} | cut -d" -f4 | sort | wc

263 263 2315
133 133 1172

$ expr 263 - 133

wc prints the numbers of lines, words, and bytes
removed 263 − 133 = 130 duplicates
Create script `getpublications.sh`:

1. `ID=$1` # The CHEBI identifier given as input is renamed to ID
2. `grep -l '<name type="scientific">Homo sapiens</name>' chebi\_\$ID\_*.xml |`
3. `xargs -I {} grep '<dbReference type="PubMed">' {} |`
4. `cut -d"" -f4 | sort -u`
Execute:

$ ./getpublications.sh 27732

How many unique publications:

$ ./getpublications.sh 27732 | wc -l

133 as expected
Complex Elements

XML elements not in the same line

use xmlint

a parser to extract data

using a XPath query

instead of single line pattern

XPath (XML Path Language)

a powerful tool to extract information from XML and HTML documents following their hierarchical structure
Namespace problems

Our protein XML files
second line defines a specific namespace
using the xmlns attribute:

```xml
```

Complicates our XPath queries
since need explicitly specify using local name
for every element in a XPath query
Get the data in each reference element:

```
$ xmllint --xpath "//*[local-name()='reference']" chebi_27732_P21817.xml
```

// means any path in the XML file
until reaching a reference element

The square brackets
normally represent conditions
Only local names

Avoid `local-name()`
identify the top-level element
extract all the data that it encloses:

$ xmllint --xpath "//*[local-name()='entry']" chebi_27732_P21817.xml
xml > chebi_27732_P21817_entry.xml

The new XML file:

```xml
<entry dataset="Swiss-Prot" created="1991-05-01" ...
<accession>P21817</accession>
...
</sequence>
</entry>
```
Apply any XPath query without explicitly say it represents a local name:

```
$ xmllint --xpath '//reference' chebi_27732_P21817_entry.xml
```

```
<reference key="1">
  <citation type="journal article" date="1990" name="J. Biol. Chem."
    volume="265" first="2244" last="2256">
    <title>Molecular cloning of cDNA encoding human and rabbit forms
      of the Ca2+ release channel (ryanodine receptor) of skeletal
      muscle sarcoplasmic reticulum.</title>
    ...
   <dbReference type="DOI" id="10.1111/cge.12810"/>
  </citation>
  <scope>VARIANTS CCD PRO-2963 AND ASP-4806</scope>
</reference>
```
Queries

//dbReference
elements of type dbReference descendants of something
<dbReference type="NCBI Taxonomy" id="9606"/>
...
<dbReference type="PubMed" id="27586648"/>

/entry//dbReference
    equivalent to the previous query
    specifying dbReference descendants of entry

/entry/reference/citation/dbReference
    similar to the previous query
    specifying the full path
    only dbReference descendants of citation
any child elements of dbReference

<property type="protein sequence ID" value="AAA60294.1"/>
...<
property type="match status" value="5"/>
//dbReference/property[1]
    first property of each dbReference
    <property type="protein sequence ID" value="AAA60294.1"/> ... <
    property type="entry name" value="MIR"/>

//dbReference/property[2]
    second property of each dbReference
    <property type="molecule type" value="mRNA"/> ... <property type=
    "match status" value="5"/>

//dbReference/property[3]
    third property of each dbReference
    <property type="molecule type" value="Genomic_DNA"/> ... <
    property type="project" value="UniProtKB"/>
//dbReference/property/@type
all type attributes of property
type="protein sequence ID" type="molecule type" type="protein sequence ID" ... type="entry name" type="match status"

//dbReference/property[@type="protein sequence ID"]
the previous property elements
with attribute type equal to protein sequence ID
<property type="protein sequence ID" value="AAA60294.1"/>
... <property type="protein sequence ID" value="ENSP00000352608"/>
//dbReference/property[@type="protein sequence ID"]/@value

string of each attribute value of previous property elements
value="AAA60294.1" value="AAC51191.1" ... value="ENSP00000352608"

/entry/sequence/text()

the contents inside sequence

MGDAEGEDEVQF...DCFRKQYEDQLS
Try previous queries:

$ xmllint --xpath '//dbReference' chebi_27732_P21817_entry.xml

Alternative o extract the PubMed identifiers:

$ xmllint --xpath '//dbReference[@type="PubMed"]/id' chebi_27732_P21817_entry.xml

... 

id="11741831"
id="16163667"
id="27586648"

Need to extract only the identifiers
Extracting XPath results

To extract the identifiers using `cut` value inside quotes

```
$ xmllint --xpath '//dbReference[@type="PubMed"]/id'
chebi_27732_P21817_entry.xml | cut -d"" -f2
```
Data Retrieval

Text Retrieval

Download the text in titles and abstracts

UniProt citations service entry

https://rest.uniprot.org/citations/1354642
link to the RDF/XML version
deal like XML

Retrieve the publication entry:

$ curl https://rest.uniprot.org/citations/1354642.rdf

Using PubMed at NCBI:

  db=pubmed&id=1354642&retmode=text&rettype=xml'
Update `getpublications.sh`:

```
1 ID=$1 # The CHEBI identifier given as input is renamed to ID
2 rm -f chebi\$_ID\_*.rdf # Removes any previous files
3 grep -l '&lt;name type="scientific">Homo sapiens&lt;/name&gt;' chebi\$_ID\_*.xml | \
   xargs -I {} grep '&lt;dbReference type="PubMed"' {} | \
5 cut -d"" -f4 | sort -u | \
6 xargs -I {} curl 'https://rest.uniprot.org/citations/{}.rdf' -o chebi\$_ID\_{}.rdf
```

Only the second and last lines updated
Execute:
$ ./getpublications.sh 27732

Take a while to download all the entries

Check files created:
$ ls chebi_27732_*.rdf

Not able to download from UniProt
Title and Abstract

title and abstract
  values of the title
  and rdfs:comment elements

Extract them:

$ xmllint --xpath "//*[local-name()='title' or local-name()='comment']" chebi_27732_1354642.rdf

<title>Polymorphisms ... hyperthermia.</title>
<rdfs:comment>Twenty-one ... gene.</rdfs:comment>
Remove the XML elements:

```bash
$ xmllint --xpath "//*[local-name()='title' or local-name()='comment']/text()" chebi_27732_1354642.rdf

Polymorphisms ... hyperthermia.
Twenty-one ... gene.
Create `gettext.sh`:

```
1 ID=$1 # The CHEBI identifier given as input is renamed to ID
2 xmllint --xpath "//*[local-name()='title' or local-name()='comment']/text()" chebi\$_ID\_*.rdf
```

Execute:

```
$ ./gettext.sh 27732 | less
```

Save the resulting text:

```
$ ./gettext.sh 27732 > chebi_27732.txt
```
Disease Recognition

Find sentences about a given disease by using `grep`:

```
$ grep 'malignant hyperthermia' chebi_27732.txt
```

To save the filtered text:

```
$ grep 'malignant hyperthermia' chebi_27732.txt > chebi_27732_hyperthermia.txt
```

Simple way of recognizing a disease
next chapters more complex text processing
Text Processing

How process that text using shell script commands specifically extract information about diseases related to caffeine

Essential set of skills to extract meaningful information from any text
Pattern Matching

Searching for mentions of *malignant hyperthermia* using related expressions:

- **MH** - acronym
- **MHS** - acronym for *malignant hyperthermia susceptible*

Solve this problem by executing:

```bash
$ grep -e 'malignant hyperthermia' -e 'MH' -e 'MHS' chebi_27732.txt
```
Case insensitive matching

Case sensitive search
  good approach to avoid wrong matches
  acronyms are normally in upper case

While full name in lowercase
  sometimes the first letter of each word
  or only the first word in uppercase
Case sensitive `grep` for the acronyms

- case insensitive `grep` for the disease words:

```bash
$ grep -e 'MH' -e 'MHS' chebi_27732.txt
$ grep -i -e 'malignant hyperthermia' chebi_27732.txt
```

Just one case sensitive `grep`

- if *Malignant hyperthermia* only alternative case:

```bash
$ grep -e 'Malignant hyperthermia' -e 'malignant hyperthermia' -e 'MH' -e 'MHS' chebi_27732.txt
```
Number of matches

Losing any match?

count number of matching lines

using the \(-c\) option:

\[$\text{grep} \ -c \ -i \ \text{'malignant hyperthermia'} \ \text{chebi}_27732.\text{txt}$\]
\[$\text{grep} \ -c \ -e \ \text{'malignant hyperthermia'} \ -e \ \text{'Malignant hyperthermia'} \ \text{chebi}_27732.\text{txt}$\]

Output should show 100 and 98 matching lines

for the insensitive and sensitive patterns

two lines not caught by the case sensitive pattern
Invert match

-v option inverts matching returns lines not matched

Get our outlier mention:

```
$ grep -i 'malignant hyperthermia' chebi_27732.txt | grep -v -e 'Malignant Hyperthermia' -e 'malignant hyperthermia'

...gene are associated with Malignant Hyperthermia (MH) and...
```

Obtain all matching lines by including missing matches:

```
$ grep -c -e 'malignant hyperthermia' -e 'Malignant hyperthermia' -e 'Malignant Hyperthermia' chebi_27732.txt
```
File Differences

diff input two files and identifies differences:

$ grep -i 'malignant hyperthermia' chebi_27732.txt > insensitive.txt
$ grep -e 'Malignant hyperthermia' -e 'malignant hyperthermia' chebi_27732.txt > sensitive.txt
$ diff sensitive.txt insensitive.txt

The output should be the same text
Problem with case sensitive matching acronyms with lowercase in middle, e.g. ChEBI

Humans not consistent mentioning acronyms in original form or all letters in uppercase or just some of them. inconsistent mentions in same publication
Evaluation metrics

Online search engines
  use case insensitive searches as default
  favor recall
while case sensitive search
favor precision
Recall the proportion of
number of correct matches
over number of correct mentions in texts (found or not found)

Case insensitive searches avoid missing mentions
so favor recall

Precision the proportion of
number of correct matches
over number of matches found (correct or incorrect)

Case sensitive searches avoid incorrect matches
so favor precision
Trade-off between precision and recall
technique improves precision decrease recall
and vice-versa

How good the trade-off is?
F-measure harmonic average of precision and recall
Acronyms appear inside common words or longer acronyms:

Searching for *MH*

the word *victimhood* matches:

```
$ echo "victimhood" | grep -i 'MH'
```

Easily solved using case sensitive matching but not longer acronym acronym *NEDMHM* for *neurodevelopmental disorder with midbrain and hindbrain malformations*:

```
$ echo "NEDMHM" | grep 'MH'
```
The `-w` option matches entire words
must be preceded and followed by
characters that are not letters, digits, or an underscore
or be at the beginning or end of the line

Neither produce a match:

```
$ echo "victimhood" | grep -w -i 'MH'
$ echo "NEDMHM" | grep -w -i 'MH'
```
Word matching improves precision but decreases recall
miss less common acronyms:

**MHE** - acronym for *malignant hyperthermia equivocal*
**MHN** - acronym for *malignant hyperthermia normal*

No match:

```
$ echo "MHE and MHN" | grep -w -i 'MH'
```

Not trivial problems to solve by exact pattern matching
Dealing with natural language text need more flexibility than exact matching

Regular expressions are an efficient tool extend exact matching with flexible patterns find different matches

Example finding all mentions of MHS or MHN regular expressions provide the alternation operator multiple alternatives to match an S or an N as the last character
Three distinct components:

- **input** - any string where to find
- **pattern** - what we are looking for
- **match** - a fragment of the input (a substring)

Input text file *chebi_27732.txt*
  - or amino acid sequences

Pattern contains special characters
  - not directly match
  - operators specify different types of matches

Matches are not replicas of the pattern
  - satisfy the specified pattern
Extended syntax

`grep` allows regular expression operators

two syntax: basic and extended

Use the extended syntax for two reasons:

- the basic not support relevant operators, e.g. alternation
- differentiate exact matching from regular expression matching
The \texttt{-E} option

not affects matching with pattern without any operator:

$ \text{echo } \text{-e } '\text{MHS}\n\text{MHN}' \mid \text{grep } '\text{MH}'$

$ \text{echo } \text{-e } '\text{MHS}\n\text{MHN}' \mid \text{grep } \text{-E } '\text{MH}'$

Use \texttt{-e} option so \texttt{echo}

interpret \texttt{\n} as a newline.

outputs two lines

grep filters lines
Alternation represented by | 
- either the preceding or following characters 
- can use parentheses specify scope

Example:
```
$ echo -e 'MHS\nMHN' | grep -E 'MH(S|N)'
```

Multiple patterns using -E option:
```
$ echo -e 'MHS\nMHN' | grep -E -e 'MH(S|X)' -e 'MH(X|N)'
```
Basic syntax

Basic syntax no match:

```bash
$ echo -e 'MHS\nMHN' | grep 'MH(S|N)' 
```

Only if and parentheses are in string:

```bash
$ echo -e 'MH(S|N)\n' | grep 'MH(S|N)' 
```
Scope

Remove the parentheses and add \texttt{--w}:

\begin{verbatim}
$ echo -e 'MHS\nMHN' | grep -w -E 'MHS|N'
\end{verbatim}

Only first line since operator applied to all preceding characters

If add a single \texttt{N} get another match:

\begin{verbatim}
$ echo -e 'MHS\nN' | grep -w -E 'MHS|N'
\end{verbatim}

Move the opening parenthesis:

\begin{verbatim}
$ echo -e 'MHS\nMHN' | grep -E 'M(HS|N)'
\end{verbatim}

Only \texttt{MHS} is now displayed
Multiple alternatives

Multiple | :
$ echo -e 'MHS\nMHN\nMHE' | grep -E 'MH(S|N|E)'

Transform previous multiple case sensitive patterns:
$ grep -c -e 'Malignant hyperthermia' -e 'Malignant Hyperthermia' -e 'malignant hyperthermia' chebi_27732.txt

in a single pattern:
$ grep -c -E '(M|m)alignant (H|h)yperthermia' chebi_27732.txt

Will obtain the same 100 matching lines
Multiple characters

Dot character (.) represents any character:

```
$ grep -o -w -E 'MH.' chebi_27732.txt | sort -u
```

-o option displays the matches, not the line

The output will be the following three-character lines:

```
MH
MH)
MH,
MH.
MH1
MH2
MHE
MHN
MHS
```
-o option counts total number of matches
not just number of lines matched:

```
$ grep -o -w -E 'MH.' chebi_27732.txt | wc -l
$ grep -c -w -E 'MH.' chebi_27732.txt
```

164 matches were found
in 47 lines
Match only the dot character use \\

```
$ grep -o -w -E 'MH\.' chebi_27732.txt | sort -u
```

Only *MH* will be displayed

Some matches are not acronyms
 e.g. *MH)* and *MH*,
Spaces

$MH$ appears because space can also be matched following text includes $MH_{\_}$, since parenthesis is word delimiter character:

... susceptible to $MH$ (MHS) ...

Text not include a word match with $MH_{\_}$:

... markers and MH susceptibility ...

Want matches where third character is letter or numerical digit
Other characters represent horizontal or vertical space
e.g. tab character
known as whitespaces
represented by \s

Both space and tab characters are matched by \s:
    echo -e 'space: :
    \ntab: \
t:' | grep -E '\s'
Groups

Group operator specify a set of characters enclosed within square brackets

Previous command replaced by:

```bash
$ echo -e 'MHS\nMHN\nMHE' | grep -E 'MH[SNE]' 
```
Ranges

Solving our need to only match letters or digits ranges with -

MH followed by any alphabet letter:

```
$ grep -o -w -E 'MH[A-Z]' chebi_27732.txt | sort -u
```

MHE
MHN
MHS
A–Z any alphabet letter in uppercase
    lowercase letter will not be matched:

```
$ echo -e 'MHS\nMHs' | grep -E 'MH[A-Z]'  
  MHS
```

Keep case sensitive grep:

```
$ echo -e 'MHS\nMHs' | grep -E 'MH[A-Za-z]'  
  MHS
  MHs
```

Dot character inside a range represents itself:

```
$ echo -e 'MHS\nMH.' | grep -E 'MH[.]'  
  MH.
```
End with a numerical digit:

```
$ grep -o -w -E 'MH[A-Z0-9]' chebi_27732.txt | sort -u
```

All three character acronyms starting with *MH*:  
- MH1
- MH2
- MHE
- MHN
- MHS
Negation

Match any character with exceptions

*MH* followed by any character except a letter

Negation feature within a group operator

represented by (*^*)

next to left bracket

all characters enclosed cannot be matched
Example:

```
$ grep -o -w -E 'MH[^A-Z]' chebi_27732.txt | sort -u
```

**MHS, MHE or MHN missing:**

- MH
- MH)
- MH,
- MH.
- MH1
- MH2
If we do not want the $MH_{\_}$ acronym, we can add the space character to the negative group:

```
$ grep -o -w -E 'MH[^A-Z ]' chebi_27732.txt | sort -u
```

The output should now contain one less acronym:

```
MH)
MH,
MH.
MH1
MH2
```
Acronyms that start with *MH* independently of their length using quantifiers operators
Optional

Item followed by ?

item can be character, operator or sub-pattern enclosed parentheses
match can either contain item or not.
Example:

```
$ grep -o -w -E 'MH[A-Z0-9]?' chebi_27732.txt | sort -u

MH
MH1
MH2
MHE
MHN
MHS
```

Third character is optional
- include two-character acronym $MH$
- not $MH_\_,$
Add space character to group:

\$ grep -o -w -E 'MH[A-Z0-9 ]?' chebi_27732.txt | sort -u

Now includes the two-character acronym \textit{MH}:

- MH
- MH
- MH
- MH1
- MH2
- MHE
- MHN
- MHS
Multiple and optional

Asterisk character *
   preceding item optional
   and be repeated multiple times

Example:

```
$ grep -o -w -E 'MH[A-Z0-9]*' chebi_27732.txt | sort -u
MH
MH1
MH2
MHE
MHN
MHS
MHS1
```
grep uses greedy approach
match as many characters as possible

Match *MH1* and not *MH*:

```
$ echo 'MH1' | grep -o -E 'MH[0-9]*'
```
Multiple and compulsory

Plus character +

Example:

```bash
$ grep -o -w -E 'MH[A-Z0-9]+' chebi_27732.txt | sort -u
```

MH1
MH2
MHE
MHN
MHS
MHS1

Not the two character acronym MH
All options

All can be reproduced by \{n, m\} where \(n\) and \(m\)
specify minimal and maximum number of occurrences
may also be omitted, no limit is imposed
Question mark ?
replaced by \{0,1\}:

Equivalent:

$ grep -o -w -E 'MH[A-Z0-9]?' chebi_27732.txt | sort -u
$ grep -o -w -E 'MH[A-Z0-9]{0,1}' chebi_27732.txt | sort -u
Asterisk character * replaced by \{0,\}
both are equivalent:

Equivalent:

```
$ grep -o -w -E 'MH[A-Z0-9]*' chebi_27732.txt | sort -u
$ grep -o -w -E 'MH[A-Z0-9]{0,}' chebi_27732.txt | sort -u
```
Plus character +
  replaced by \{1,\}

Equivalent:
$ grep -o -w -E 'MH[A-Z0-9]+' chebi_27732.txt | sort -u
$ grep -o -w -E 'MH[A-Z0-9]{1,}' chebi_27732.txt | sort -u
Using \( \{1,1\} \)

same as not having any operator.

both are equivalent:

\[
\text{
$\text{grep -o -w -E 'MH[A-Z0-9]' chebi_27732.txt | sort -u}$}
\]

\[
\text{
$\text{grep -o -w -E 'MH[A-Z0-9]{1,1}' chebi_27732.txt | sort -u}$}
\]

MH1
MH2
MHE
MHN
MHS
Acronyms with exactly 4 characters:

$ grep -o -w -E 'MH[A-Z0-9]{2,2}' chebi_27732.txt | sort -u

MHS1
Matches specific parts of input, examples:
- identify start and stop codons in sequence
- lines starting with a name of a disease

Regular expressions patterns can:
- start with ^
- end with $
Lines starting with **Malignant Hyperthermia**:  
$\text{grep \ -E \ '^(M|m)alignant \ (H|h)yperthermia' \ chebi_27732.txt}$

...  
Malignant hyperthermia (MH) is a potentially fatal autosomal ...  
Malignant hyperthermia (MH) is a pharmacogenetic disorder ...
Check how many lines filtered:

\[
\begin{align*}
\text{"grep \ -c\ -E\ '^(M|m)alignant\ (H|h)yperthermia'\ chebi_27732.txt"} \\
\text{"grep \ -c\ -E\ '(M|m)alignant\ (H|h)yperthermia'\ chebi_27732.txt"}
\end{align*}
\]

Only 20 of the 100 matches were considered
Ending

Lines ending with *Malignant Hyperthermia*,

```bash
$ grep -E '(M|m)alignant (H|h)yperthermia.$' chebi_27732.txt
```

... Mutations in the ryanodine receptor gene in central core disease and malignant hyperthermia. Detection of a novel mutation at amino acid position 614 in the ryanodine receptor in malignant hyperthermia. Novel mutations at a CpG dinucleotide in the ryanodine receptor in malignant hyperthermia.

Allow a punctuation character before the end of the line
added the dot before the dollar
Check how many lines filtered:

```
$ grep -c -E '(M|m)alignant (H|h)yperthermia.$' chebi_27732.txt
$ grep -c -E '(M|m)alignant (H|h)yperthermia' chebi_27732.txt
```

Only 15 of the 100 matches were at the end of the line
Near the end

Mention not ending exactly at the last character
    allow a following expression
    or a given number of characters

Allow 10 other characters:

$ grep -c -E '(M|m)alignant (H|h)yperthermia.{0,10}$' chebi_27732.txt

The output will show that we have 20 matches.
Remove $-c$ and check

*families* and *patients* are now allowed

... Novel mutations in C-terminal channel region of the ryanodine receptor in malignant hyperthermia patients.

...

Novel missense mutations and unexpected multiple changes of RYR1 gene in 75 malignant hyperthermia families.

...
Word in between

Allow a word in between
independently of its length
optional sequence of non-space characters (the word)
preceded by a space:

```
$ grep -c -E '(M|m)alignant (H|h)yperthermia( [^ \]*)?.$'
  chebi_27732.txt
```

Only 24 matches

[^ ] operator avoids having two words
Remove –c and check lengthy words (with more than 10 characters) such as susceptibility are now allowed

... Ryanodine receptor gene point mutation and malignant hyperthermia susceptibility.

...
Lines start with *Malignant Hyperthermia*
and end with an acronym:

```
$ grep -E '^\(M|m\)alignant (H|h)yperthermia' chebi_27732.txt | grep -w -E 'MHS?.$'
```

Or add both the circumflex and dollar operators:

```
$ grep -w -E '^\(M|m\)alignant (H|h)yperthermia.*MHS?.$' chebi_27732.txt
```

.* to match anything in between them
Match all the text of the abstract each abstract in a single line:
Malignant hyperthermia (MH) is a pharmacogenetical complication ... as for genetic diagnosis of MH.
Malignant hyperthermia susceptibility (MHS) is a subclinical pharmacogenetic disorder ... been tested positive for MHS.

Problem of tokenization
need to match a full sentence or a phrase
Match position

Knowing exact position of matches
using \texttt{-b}

\begin{verbatim}
$ echo 'MHS MHN MHE' | grep -b -o -w -E 'MH[SNE]'

  0:MHS
  4:MHN
  8:MHE
\end{verbatim}

Same result in multiple lines:

\begin{verbatim}
$ echo -e 'MHS\nMHN\nMHE' | grep -b -o -w -E 'MH[SNE]'
\end{verbatim}
Tokenization

Work at the level of a sentence
not use a full document as the input string

Tokenization is a Natural Language Processing (NLP) task
identifying boundaries in the text
to fragment it into basic units called tokens
sentences, phrases, multi-word expressions, or words.
Character delimiters

Specific characters as accurate boundaries to fragment text into tokens.
  space character to identify words
  . ? ! to identify ending of sentence
  , ; : parenthesis to identify a phrase

More complex in languages without explicitly delimiters such as Chinese
Replace these delimiters by newline characters result in a token per line:

$ tr '![.!?]' '\n' < chebi_27732.txt | wc -l

Get 1618 lines from the original 255 lines:

$ wc -l chebi_27732.txt
Wrong tokens

Not so simple, analyze:

```bash
$ tr '[.!?]' '\n' < chebi_27732.txt | less
```

i) many lines are empty
   extra newline character added to last sentence

ii) the dot character also used as decimal mark
    sentences split in multiple lines by having decimal numbers
Example:

These 10 mutations account for 21.9% of the North American MH-susceptible population

Split in two lines:

These 10 mutations account for 21.9% of the North American MH-susceptible population
String Replacement

One character not enough, need context

`sed` powerful version of `tr`
stream editor receive as input a string
perform basic text transformations
replace one expression by another
Replace *caffeine* by its ChEBI identifier:

```
$ sed -E 's/caffeine/CHEBI:27732/gi' chebi_27732.txt
```

's/FIND/REPLACE/FLAGS'

**FIND** pattern to find

**REPLACE** the expression to replace

**FLAGS** multiple options:

- `g` replace all matches not just the first in line
- `i` case insensitive.

- `E` use extended regular expressions
Original fragment of text:

... link between the caffeine threshold and tension ...

converted to:

... link between the CHEBI:27732 threshold and tension ...
Multi-character delimiters

Replace delimiter characters by a newline when followed by at least one space:

```bash
$ sed -E 's/\[.!?] +/\n/g' chebi_27732.txt
```

Avoids empty lines
not splits sentence in end of line, assuming no ghost spaces

Preserves decimal markers
followed by numerical digits, not spaces.

Get 1618 lines from original 255 lines:

```bash
$ sed -E 's/\[.!?] +/\n/g' chebi_27732.txt | wc -l
```
Keep delimiters

Previous `sed` removes delimiter characters
  may cause other problems
  better solution is keep them

`sed` allows keep each match
  sub-pattern enclosed within parentheses
  and use backslash and its numerical order
$ \texttt{sed -E 's/([.!?])( +)/\1\n\2/g' chebi_27732.txt}$

**From:**

... muscle relaxants. To date, ...

**To:**

... muscle relaxants.
To date, ...
Some sentences include a HTML elements:

... bulk.&lt;h4&gt;Methods&lt;/h4&gt;Fetal ...
... sequencing.&lt;h4&gt;Results&lt;/h4&gt;Whole ...
Add the option of & character:

```
$ sed -E 's/([.!?])([& ]+)/\1\n\2/g' chebi_27732.txt | wc -l
```

Get 1179 lines
able to split more 87 sentences.

Not free of errors
almost impossible to derive a rule
that covers all the possible typos humans produce
I watch three climb before it’s my turn. It’s a tough one. The guy before me tries twice. He falls twice. After the last one, he comes down. He’s finished for the day. It’s my turn. My buddy says “good luck!” to me. I noticed a bit of a problem. There’s an outcrop on this one. It’s about halfway up the wall. It’s not a

Pattern equivalent to \. \{2,\} [A-Z] identifies multiples spaces at the beginning of a sentence

(Adapted from: https://en.wikipedia.org/wiki/Regular_expression)
**Sentences file**

Update *gettext.sh*:

1. ID=$1 # The CHEBI identifier given as input is renamed to ID
2. `xmllint --xpath "//*[local-name()='title' or local-name()='comment']/text()" chebi\_${ID}\_*.rdf |` 
3. `sed -E 's/([^!?.])([^ ]+)/\1
\2/g'

Save output:

```
$ ./gettext.sh 27732 > chebi_27732_sentences.txt
```

Each line is now a sentence
Entity recognition

Select sentences with acronyms:

$ grep -w -E 'MH[SNE]?' chebi_27732_sentences.txt

... Interestingly, the data suggest a link between the caffeine threshold and tension values and the MH/CCD phenotype.
Use \texttt{-n} get the number of line:

\begin{verbatim}
$ grep -n -o -w -E 'MH[SNE]?' chebi_27732_sentences.txt
\end{verbatim}

\begin{verbatim}
...  
1129:MH  
1131:MH  
1132:MH
\end{verbatim}
Add the \texttt{-b} option:

\begin{verbatim}
$ grep -b -n -o -w -E 'MH[SNE]?' chebi_27732_sentences.txt
\end{verbatim}

Number of the line, the character position, and the match:

\begin{verbatim}
\end{verbatim}
Script receives pattern as argument
the input text as the standard input,
display the line numbers and the matches
in a TSV format

Create script `getentities.sh`:

```bash
1  PATTERN=$1
2  grep -n -o -w -E $PATTERN | \
3  tr ':' '	'
```

First line stores the pattern
`grep` finds the matches
`tr` replaces each colon by tab
Execute:

$ ./getentities.sh 'MH[SNE]?' < chebi_27732_sentences.txt

...  
1129  MH  
1131  MH  
1132  MH

Values separated by tab (TSV format)
Saved as a TSV file

open in spreadsheet application:

```
$ ./getentities.sh 'MH[SNE]?' < chebi_27732_sentences.txt > chebi_27732.tsv
```
Select the sentence

Analyze a specific matched sentence
text editor go to that line number

Or use `p` option of `sed`
output a given line number

```bash
$ sed -n '2p' chebi_27732_sentences.txt

... in susceptible people (MHS) by volatile ...
```
Pattern File

Recognize different entities
or different mentions of same entity
the official name, synonyms, and acronyms

grep allows list of patterns from file
using –f option
Create a text file `patterns.txt`:

(M|m)alignant (H|h)yperthermia
MH[SNE]?
(C|c)affeine

Execute:

```
$ grep -n -o -w -E -f patterns.txt chebi_27732_sentences.txt

... 1131:caffeine 1132:caffeine 1132:MH
```
Update *getentities.sh*:

1. `PATTERNS=$1`
2. `grep -n -o -w -E -f $PATTERNS | \
3. `tr '[:']' '\t'`

Execute:

```
$ ./getentities.sh patterns.txt < chebi_27732_sentences.txt
```

Save output:

```
$ ./getentities.sh patterns.txt < chebi_27732_sentences.txt > chebi_27732.tsv
```
patterns.txt useful
not focused in a single disease
find any disease mentioned

Create file with the full lexicon of diseases
addressed in the following chapter
Relation Extraction

Sentences describe possible relationships
e.g. disease and compound.
complex text mining challenge

Simple approach:

```
$ grep -n -w -E 'MH[SNE]?.*(C|c)affeine' chebi_27732_sentences.txt
```

257: ... MHS families were investigated with a caffeine ...

One of the eight displayed sentences
Missing all with *caffeine* first:

```
$ grep -n -w -E '(C|c)affeine.*MH[SNE]?' chebi_27732_sentences.txt
```

837: ... caffeine–halothane contracture test were greater in those who had a known MH ...

1132: ... caffeine threshold and tension values and the MH ...

Multiple filters

Most flexible approach two `grep`

first selects sentences mentioning one entity
the other selects from the previously selected sentences
the ones mentioning the other entity:

```
$ grep -n -w -E 'MH[SNE]?' chebi_27732_sentences.txt | grep -w -E '(C|c)affeine'
```

Shows ten sentences mentioning `caffeine` and an acronym
Relation type

Specific type of relationship
  additional filter for specific verb

Example filter with *response* or *diagnosed*:

```
$ grep -n -w -E 'MH\[SNE]\?' chebi_27732_sentences.txt | grep -w -E '(C|c)affeine' | grep -w -E 'response|diagnosed'
```
Not take in account where the verb appears

*response* appears first than the two entities:

58: The relationship between the IVCT response and genotype was
... the number of MHS discordants ... at 2.0 mM caffeine ...

Between the two entities:

$ grep -n -w -E 'MH[SNE]?.*(response|diagnosed).*(C|c)affeine' chebi_27732_sentences.txt

Previous sentence not a match
Remove relation types

Filtering specific type of relations

use `-v`:

```bash
$ grep -n -w -E 'MH\[SNE\]?' chebi_27732_sentences.txt | grep -w -E '(C|c)affeine' | grep -v -w -E 'response|diagnosed'
```

Resulting sentences not mention *response* or *diagnosed*
Semantic Processing

Introduce the world of semantics
retrieve and enhance text and data processing
by using semantics

Explore semantic resources
nowadays available
Classes

Searched for *caffeine* and *malignant hyperthermia*
miss related entities
can be found in semantic resources
such as ontologies.

Semantics of *caffeine* and *malignant hyperthermia*
in *ChEBI* and *DO* ontologies
Retrieving both ontologies (OWL files):

```bash
$ curl -L -O http://purl.obolibrary.org/obo/doid/releases
   /2021-03-29/doid.owl
$ curl -L -O http://purl.obolibrary.org/obo/chebi/198/chebi_lite.owl
```

-o saves to file name as remote file (last part of URL)
files `chebi_lite.owl` and `doid.owl`
-L follows redirects

Most recent release:

http://purl.obolibrary.org/obo/doid.owl
http://purl.obolibrary.org/obo/chebi/chebi_lite.owl
Class label

OWL files use XML syntax

check entities:

$ grep '>malignant hyperthermia<' doid.owl
$ grep '>caffeine<' chebi_lite.owl

<rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">malignant hyperthermia</rdfs:label>
<rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">caffeine</rdfs:label>

Property label (*rdfs:label*),
inside class definition
Class definition

Retrieve the full class definition with `xmllint`:

```
$ xmllint --xpath "//*[local-name()='label' and text()='malignant hyperthermia']/.." doid.owl
```

The XPath query

- find the label *malignant hyperthermia*
- then .. the parent element, *Class element*
Semantics of *malignant hyperthermia*

much more than its label:

<owl:Class rdf:about="http://purl.obolibrary.org/obo/DOID_8545">
  <rdfs:subClassOf rdf:resource="http://purl.obolibrary.org/obo/DOID_0050736"/>
  <rdfs:subClassOf rdf:resource="http://purl.obolibrary.org/obo/DOID_66"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="http://purl.obolibrary.org/obo/IDO_0000664"/>
      <owl:someValuesFrom rdf:resource="http://purl.obolibrary.org/obo/GENO_0000147"/>
    </owl:Restriction>
  </rdfs:subClassOf>
...
  <oboInOwl:hasExactSynonym xml:lang="en">anesthesia related hyperthermia</oboInOwl:hasExactSynonym>
  <oboInOwl:hasExactSynonym xml:lang="en">malignant hyperpyrexia due to anesthesia</oboInOwl:hasExactSynonym>
  <oboInOwl:hasOBONamespace rdf:datatype="http://www.w3.org/2001/
XMLSchema#string">disease_ontology</oboInOwl:hasOBONamespace>
<oboInOwl:id rdf:datatype="http://www.w3.org/2001/XMLSchema#string">DOID:8545</oboInOwl:id>
<oboInOwl:inSubset rdf:resource="http://purl.obolibrary.org/obo/doid#DO_rare_slim"/>
<oboInOwl:inSubset rdf:resource="http://purl.obolibrary.org/obo/doid#NCIthesaurus"/>
<rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Xref MGI.
OMIM mapping confirmed by DO. [SN].</rdfs:comment>
<rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">malignant hyperthermia</rdfs:label>
</owl:Class>
Class description of *malignant hyperthermia* in the Human Disease Ontology
malignant hyperthermia subclass of (specialization)
  entries 0050736 autosomal dominant disease
  and 66 muscle tissue disease

malignant hyperthermia a special case of muscle tissue disease
  and of autosomal dominant disease
Search relations with 0050736 and 66:

```
$ xmllint --xpath "//*[local-name()='label' and text()='malignant hyperthermia']/..//*[local-name()='resource' and .='http://purl.obolibrary.org/obo/DOID_0050736' or .='http://purl.obolibrary.org/obo/DOID_66']" doid.owl
```

@*[local-name()=']resource']

extracts the URI specified in an attribute resource of any descendant //*[...].

Uses the subClassOf element:

```
<rdfs:subClassOf rdf:resource="http://purl.obolibrary.org/obo/DOID_0050736"/>
<rdfs:subClassOf rdf:resource="http://purl.obolibrary.org/obo/DOID_66"/>
```
Retrieve full class definition of caffeine:

```
$ xml2xquery --xpath "//*[local-name()='label' and text()='caffeine']/.." chebi_lite.owl
```

Semantics of caffeine differs from malignant hyperthermia still share many properties such as subClassOf
A trimethylxanthine in which the three methyl groups are located at positions 1, 3, and 7. A purine alkaloid that occurs naturally in tea and coffee.
<oboInOwl:hasAlternativeId rdf:datatype="http://www.w3.org/2001/XMLSchema#string">CHEBI:22982</oboInOwl:hasAlternativeId>
<oboInOwl:hasAlternativeId rdf:datatype="http://www.w3.org/2001/XMLSchema#string">CHEBI:3295</oboInOwl:hasAlternativeId>
<oboInOwl:hasAlternativeId rdf:datatype="http://www.w3.org/2001/XMLSchema#string">CHEBI:41472</oboInOwl:hasAlternativeId>
<oboInOwl:hasOBONamespace rdf:datatype="http://www.w3.org/2001/XMLSchema#string">chebi_ontology</oboInOwl:hasOBONamespace>
<oboInOwl:id rdf:datatype="http://www.w3.org/2001/XMLSchema#string">CHEBI:27732</oboInOwl:id>
<oboInOwl:inSubset rdf:resource="http://purl.obolibrary.org/obo/chebi#3_STAR"/>
<rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">caffeine</rdfs:label>
</owl:Class>
Semantic Processing

Class: caffeine

Term IRI: http://purl.obolibrary.org/obo/CHEBI_27732

Definition: A triethylxanthine in which the three methyl groups are located at positions 1, 3, and 7. A purine alkaloid that occurs naturally in tea and coffee.

Annotations


- has_exact_synonym: CAFFEINE; caffeine; 1,3,7-trimethyl-3,7-dihydro-1H-purine-2,6-dione; caffeine

- has_oobo_namespace: chebi_ontology

- has_related_synonym: The7; guanine; caffeine; the7methylthecobrine; 1,3,7-trimethyl-2,6-dioxopurine; 3,7-dihydro-1,3,7-trimethyl-1H-purine-2,6-dione; 1,3,7-trimethylxanthine; adenosine; caffeine; 1,3,7-trimethylxanthine; 7-methylxanthine; Caffeine; caffeine; 1,3,7-trimethylxanthine; 2,6-dione; mateine; methylthecobrine; Koffein; Kofein; http://purl.obolibrary.org/obo/chebi/charge:0

- http://purl.obolibrary.org/obo/chebi/formula: C8H10N4O2

- http://purl.obolibrary.org/obo/chebi/inchi: iInChI=1S/C8H10N4O2c1-10-4-9-5-9[10](7)[13](12)(3)(6)(14)[11](6)2h4H1\-3H0


- http://purl.obolibrary.org/obo/chebi/molecularFormula: C8H10N4O2


- In_subset: http://purl.obolibrary.org/obo/chebi#5_STAR

Class description of caffeine in ChEBI

(Source: http://www.ontobee.org/)
caffeine specialization of

26385 \textit{purine alkaloid} and 27134 \textit{trimethylxanthine}

Search relations:

```
$ xmllint --xpath "//*[local-name()='label' and text()='caffeine']/..//*[@*[local-name()='resource' and .='http://purl.obolibrary.org/obo/CHEBI_26385' or .='http://purl.obolibrary.org/obo/CHEBI_27134']]]" doid.owl
```

```
<rdfs:subClassOf rdf:resource="http://purl.obolibrary.org/obo/CHEBI_26385"/>
<rdfs:subClassOf rdf:resource="http://purl.obolibrary.org/obo/CHEBI_27134"/>
```
Related Classes

Superclasses & Asserted Axioms

- has material basis in some autosomal dominant inheritance
- muscle tissue disease
- autosomal dominant disease

Related classes of *malignant hyperthermia* in the Human Disease Ontology

(Source: [http://www.ontobee.org/](http://www.ontobee.org/))
### Semantic Processing Classes

**Superclasses & Asserted Axioms**

- has role some plant metabolite
- has role some fungal metabolite
- has role some environmental contaminant
- has role some human blood serum metabolite
- has role some food additive
- has role some ryanodine receptor agonist
- has role some adenosine receptor antagonist
- has role some mouse metabolite
- has role some EC 3.1.4.1 (phosphoric diester hydrolase) inhibitor
- has role some EC 2.7.11.1 (non-specific serine/threonine protein kinase) inhibitor
- has role some adenosine A2A receptor antagonist
- has role some adjuvant
- has role some central nervous system stimulant
- has role some psychotrophic drug
- has role some diuretic
- has role some xenobiotic
- has role some mutagen
- has role some purine alkaloid
- has role some trimethylxanthine

---

**Related classes of caffeine in ChEBI**

(Source: [http://www.ontobee.org/](http://www.ontobee.org/))

---

Data and Text Processing for Health and Life Sciences by F. Couto  
Slide 293 of 403
Search for relation with *mutagen* (25435)

```bash
$ xmllint --xpath "//*[local-name()='label' and text()='caffeine']/..//*[@*[local-name()='resource' and .='http://purl.obolibrary.org/obo/CHEBI_25435']]/../.." chebi_lite.owl
```

Defined by RO:0000087 (*has role*) in *Relations Ontology*:

```xml
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resource="http://purl.obolibrary.org/obo/RO_0000087"/>
    <owl:someValuesFrom rdf:resource="http://purl.obolibrary.org/obo/CHEBI_25435"/>
  </owl:Restriction>
</rdfs:subClassOf>
```
Search *has role* in OWL:

```
$ xmllint --xpath "//*[local-name()='ObjectProperty'] [@*[local-name()='about'] = 'http://purl.obolibrary.org/obo/RO_0000087']" chebi_lite.owl
```

Finds *ObjectProperty*

selects the ones with *about* attribute

with the relation URI as value.
Neither transitive or cyclic:

```
<owl:ObjectProperty rdf:about="http://purl.obolibrary.org/obo/RO_0000087">
  ...
  <oboInOwl:id rdf:datatype="http://www.w3.org/2001/XMLSchema#string">has_role</oboInOwl:id>
  <oboInOwl:is_cyclic rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">false</oboInOwl:is_cyclic>
  <oboInOwl:is_transitive rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">false</oboInOwl:is_transitive>
  ...
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">has role</rdfs:label>
</owl:ObjectProperty>
```
ObjectProperty: has role

Term IRI: http://purl.obolibrary.org/obo/RO_0000087

Definition: A relation between an independent continuant (the bearer) and a role, in which the role specifically depends on the bearer for its existence

Annotations

- editor note: A bearer can have many roles, and its roles can exist for different periods of time, but none of its roles can exist when the bearer does not exist. A role need not be realized at all the times that the role exists.
- alternative term: has_role
- example of usage: this person has role this investigator role (more colloquially: this person has this role of investigator)

Property Hierarchy

1. topObjectProperty
   - bearer of
     - has function
     - has qualify
     - has disposition
     - has role
       - has biological role
       - has application role
       - has chemical role

Description of has role property

(Source: http://www.ontobee.org/)
URIs and Labels

Standardize the process
   scripts convert label into URI
   and vice-versa

Internal ontology processing using URIs
   then convert to labels
URI of a label

Get URI of *malignant hyperthermia*:

```bash
$ xmllint --xpath "//*[local-name()='label' and text()='malignant hyperthermia']/../@*[local-name()='about']" doid.owl

rdf:about="http://purl.obolibrary.org/obo/DOID_8545"
```
Only the value, add `string`:

```bash
$ xmlindent --xpath "string(/*[local-name()='label' and text()='malignant hyperthermia']/../@*[local-name()='about'])" doid.owl
http://purl.obolibrary.org/obo/DOID_8545
```

`string` returns only one attribute value even if many are matched assuming `malignant hyperthermia` is unambiguous
Alternative:

**add a cut to get the URI**

```bash
$ xmlllint --xpath "//*[local-name()='label' and text()='malignant hyperthermia']/../@*[local-name()='about']" doid.owl | cut -d" -f2
$ xmlllint --xpath "//*[local-name()='label' and text()='caffeine']/../@*[local-name()='about']" chebi_lite.owl | cut -d" -f2
```
Script `geturi.sh`:

```bash
1  OWLFILE=$1
2  xargs -I {} xmllint --xpath "//*[local-name()='label' and text()='{}']/../@*[local-name()='about']" $OWLFILE | \
3  cut -d" -f2
```

Multiple labels as standard input
OWL file to find URIs as argument
Execute:

$ echo 'malignant hyperthermia' | ./geturi.sh doid.owl
$ echo 'caffeine' | ./geturi.sh chebi_lite.owl

http://purl.obolibrary.org/obo/DOID_8545
http://purl.obolibrary.org/obo/CHEBI_27732
Execute using multiple labels:

```
$ echo -e 'malignant hyperthermia\nmuscle tissue disease' | ./geturi.sh doid.owl
$ echo -e 'caffeine\npurine alkaloid\ntrimethylxanthine' | ./geturi.sh chebi_lite.owl
```

http://purl.obolibrary.org/obo/DOID_8545
http://purl.obolibrary.org/obo/DOID_66

http://purl.obolibrary.org/obo/CHEBI_27732
http://purl.obolibrary.org/obo/CHEBI_26385
http://purl.obolibrary.org/obo/CHEBI_27134
Label of a URI

Get label disease 8545:

$ xmllint --xpath "//*[local-name()='Class'][@*[local-name()='about'][local-name()='http://purl.obolibrary.org/obo/DOID_8545']/*[local-name()='label']/text()" doid.owl

malignant hyperthermia

@*[local-name()='label']
selects element describes label

Problem if multiple matches

text() all labels in same line
alternative add tr and grep
Get label of compound 27732:

```
$ xmllint --xpath "//*[local-name()='Class'][@@name='about'] = 'http://purl.obolibrary.org/obo/CHEBI_27732']/*[local-name()='label']/text()" chebi_lite.owl
```

caffeine
Script `getlabels.sh`:

1. `OWLFILE=$1`
2. `xargs -I {} xmllint --xpath "//*[local-name()='Class'][@*[local-name()='about']='{$'}]/*[local-name()='label']/text()" $OWLFILE`

Multiple URIs as standard input

OWL file to find labels as argument

`xargs` process each line of standard input
Execute:

$ echo 'http://purl.obolibrary.org/obo/DOID_8545' | ./getlabels.sh
doid.owl

$ echo 'http://purl.obolibrary.org/obo/CHEBI_27732' | ./getlabels.sh
chebi_lite.owl

malignant hyperthermia
caffeine
Execute with multiple URIs:

$ echo -e 'http://purl.obolibrary.org/obo/DOID_8545
http://purl.obolibrary.org/obo/DOID_66' | ./getlabels.sh doid.owl

$ echo -e 'http://purl.obolibrary.org/obo/CHEBI_27732
http://purl.obolibrary.org/obo/CHEBI_26385
http://purl.obolibrary.org/obo/CHEBI_27134' | ./getlabels.sh chebi_lite.owl

malignant hyperthermia
muscle tissue disease

caffeine
purine alkaloid
trimethylxanthine
Test both scripts:

```bash
$ echo -e 'malignant hyperthermia
muscle tissue disease' | ./
ascimento.sh doid.owl | ./getlabels.sh doid.owl
$ echo -e 'caffeine
purine alkaloid
trimethylxanthine' | ./geturi
ascimento.sh chebi_lite.owl | ./getlabels.sh chebi_lite.owl
```

malignant hyperthermia
muscle tissue disease

caffeine
purine alkaloid
trimethylxanthine
**URIs as input:**

$ echo -e 'http://purl.obolibrary.org/obo/DOID_8545
http://purl.obolibrary.org/obo/DOID_66' | ./getlabels.sh doid.owl | ./geturi.sh doid.owl

$ echo -e 'http://purl.obolibrary.org/obo/CHEBI_27732
http://purl.obolibrary.org/obo/CHEBI_26385
http://purl.obolibrary.org/obo/CHEBI_27134' | ./getlabels.sh chebi_lite.owl | ./geturi.sh chebi_lite.owl

- http://purl.obolibrary.org/obo/DOID_8545
- http://purl.obolibrary.org/obo/DOID_66
- http://purl.obolibrary.org/obo/CHEBI_27732
- http://purl.obolibrary.org/obo/CHEBI_26385
- http://purl.obolibrary.org/obo/CHEBI_27134
Synonyms

Not always mentioned using official label
text alternative labels
represented by hasExactSynonym

Synonyms of a disease:

$ xmllint --xpath "//*[local-name()='Class'] [@*[local-name()='about'][='http://purl.obolibrary.org/obo/DOID_8545']/*[local-name()='hasExactSynonym']" doid.owl

<oboInOwl:hasExactSynonym rdf:datatype="http://www.w3.org/2001/XMLSchema#string">anesthesia related hyperthermia</oboInOwl:hasExactSynonym>

<oboInOwl:hasExactSynonym rdf:datatype="http://www.w3.org/2001/XMLSchema#string">malignant hyperpyrexia due to anesthesia</oboInOwl:hasExactSynonym>
Both primary label and synonyms:

```bash
$ xml lint --xpath "//*[local-name()='Class'][@*['@local-name()']=='about'='http://purl.obolibrary.org/obo/DOID_8545']/*[local-name()=='hasExactSynonym' or local-name()='label']" doid.owl
```

```xml
<oboInOwl:hasExactSynonym rdf:datatype="http://www.w3.org/2001/XMLSchema#string">anesthesia related hyperthermia</oboInOwl:hasExactSynonym>
<oboInOwl:hasExactSynonym rdf:datatype="http://www.w3.org/2001/XMLSchema#string">malignant hyperpyrexia due to anesthesia</oboInOwl:hasExactSynonym>
<rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">malignant hyperthermia</rdfs:label>
```
Update `getlabels.sh`:

1. `OWLFILE=$1`
2. `xargs -I {} xmllint --xpath "//*[local-name()='Class'][@*[local-name()='about'='{}']/*[local-name()='hasExactSynonym' or local-name()='hasRelatedSynonym' or local-name()='label']/text()" $OWLFILE`

Adding the `hasExactSynonym` keyword
and `hasRelatedSynonym`
Execute:

```bash
$ echo -e 'http://purl.obolibrary.org/obo/DOID_8545' | ./getlabels.sh doid.owl

anesthesia related hyperthermia
malignant hyperpyrexia due to anesthesia
malignant hyperthermia
```
Send output to `geturi.sh`:

```
$ echo 'http://purl.obolibrary.org/obo/DOID_8545' | ./getlabels.sh
doid.owl | ./geturi.sh doid.owl
```

**XPath warnings for the two synonyms:**

```
XPath set is empty
XPath set is empty
http://purl.obolibrary.org/obo/DOID_8545
```
Ignore these mismatches:

```bash
$ echo 'http://purl.obolibrary.org/obo/DOID_8545' | ./getlabels.sh
doid.owl | ./geturi.sh doid.owl 2>/dev/null
```

Or update `geturi.sh` to include synonyms:

```bash
1 OWLFILE=$1
2 xargs -I {} xmllint --xpath "//*[local-name()='hasExactSynonym'
    or local-name()='hasRelatedSynonym' or local-name()='label'
    and text()='{}']/../@*[local-name()='about']" $OWLFILE | \
3 cut -d"" -f2
```
Execute:

```
$ echo 'http://purl.obolibrary.org/obo/DOID_8545' | ./getlabels.sh
doid.owl | ./geturi.sh doid.owl

http://purl.obolibrary.org/obo/DOID_8545
http://purl.obolibrary.org/obo/DOID_8545
http://purl.obolibrary.org/obo/DOID_8545
http://purl.obolibrary.org/obo/DOID_8545
```
Avoid repetitions:

```
$ echo 'http://purl.obolibrary.org/obo/DOID_8545' | ./getlabels.sh doid.owl | ./geturi.sh doid.owl | sort -u
http://purl.obolibrary.org/obo/DOID_8545
```
Parent classes of *malignant hyperthermia*:

```
$ xmllint --xpath "/*/[@*['about']='http://purl.obolibrary.org/obo/DOID_8545']/*[@*['subClassOf']]/@*[local-name()='resource']" doid.owl
```

[local-name()='subClassOf'] gets subclass

@*[local-name()='resource'] gets attribute with URI

**Output URIs parents of 8545:**

```
<rdf:resource="http://purl.obolibrary.org/obo/DOID_0050736"/>
<rdf:resource="http://purl.obolibrary.org/obo/DOID_66"
```
Execute for *caffeine*:

```bash
$ xmllint --xpath "//*[local-name()='Class'][@*[local-name()='about']='http://purl.obolibrary.org/obo/CHEBI_27732']/*[local-name()='subClassOf']/*[local-name()='resource']" chebi_lite.owl
	rdf:resource="http://purl.obolibrary.org/obo/CHEBI_26385"
	rdf:resource="http://purl.obolibrary.org/obo/CHEBI_27134"
```

No longer can use `string`

*multiple parents*

and `string` only returns first match
Get only URIs:

```bash
$ xmllint --xpath "//*[local-name()='Class'][*[@local-name()='about'='http://purl.obolibrary.org/obo/CHEBI_27732']/*[local-name()='subClassOf']/[@local-name()='resource']" chebi_lite.owl | cut -d" -f2

http://purl.obolibrary.org/obo/CHEBI_26385
http://purl.obolibrary.org/obo/CHEBI_27134
```
**Script `getparents.sh`:**

```bash
1  OWLFILE=$1
2  xargs -I {} xmllint --xpath "//*[local-name()='Class'][@*[local-name()='about']='{}'/*[local-name()='subClassOf']/@*[local-name()='resource']"] $OWLFILE | \ 
3  cut -d"" -f2
```

**Multiple URIs given as standard input**

**OWL file to find parents as argument**
Parents of *malignant hyperthermia*:

```bash
$ echo 'http://purl.obolibrary.org/obo/DOID_8545' | ./getparents.sh
doid.owl

http://purl.obolibrary.org/obo/DOID_0050736
http://purl.obolibrary.org/obo/DOID_66
Labels of parents

Redirect the output:

```bash
$ echo 'http://purl.obolibrary.org/obo/DOID_8545' | ./getparents.sh
doid.owl | ./getlabels.sh doid.owl

autosomal dominant disease
muscle tissue disease
```
Same with caffeine:

```bash
$ echo 'http://purl.obolibrary.org/obo/CHEBI_27732' | ./getparents.sh chebi_lite.owl | ./getlabels.sh chebi_lite.owl
```

- purine alkaloid
- trimethylxanthine
Related classes

All related classes besides \textit{subClassOf}:

\begin{verbatim}
$ xmllint --xpath "//*[local-name()='Class'][@*[local-name()='about']
'='http://purl.obolibrary.org/obo/CHEBI_27732']/*[local-name()='subClassOf']//*[local-name()='someValuesFrom']/*@*[local-name()=
'resource']" chebi_lite.owl | cut -d" -f2
\end{verbatim}

Related classes are in attribute \textit{resource}
of \textit{someValuesFrom} element inside \textit{subClassOf} element
Related classes of caffeine:

http://purl.obolibrary.org/obo/CHEBI_25435
http://purl.obolibrary.org/obo/CHEBI_35337
http://purl.obolibrary.org/obo/CHEBI_35471
http://purl.obolibrary.org/obo/CHEBI_35498
http://purl.obolibrary.org/obo/CHEBI_35703
http://purl.obolibrary.org/obo/CHEBI_38809
http://purl.obolibrary.org/obo/CHEBI_50218
http://purl.obolibrary.org/obo/CHEBI_50925
http://purl.obolibrary.org/obo/CHEBI_53121
http://purl.obolibrary.org/obo/CHEBI_60809
http://purl.obolibrary.org/obo/CHEBI_64047
http://purl.obolibrary.org/obo/CHEBI_67114
http://purl.obolibrary.org/obo/CHEBI_71232
http://purl.obolibrary.org/obo/CHEBI_75771
http://purl.obolibrary.org/obo/CHEBI_76924
http://purl.obolibrary.org/obo/CHEBI_76946
http://purl.obolibrary.org/obo/CHEBI_78298
http://purl.obolibrary.org/obo/CHEBI_85234
Labels of related classes

Add `getlabels.sh`:

```bash
$ xmllint --xpath "//*[local-name()='Class'][@*[local-name()=='about']
'='http://purl.obolibrary.org/obo/CHEBI_27732']/*[local-name()=
='subClassOf']/*[local-name()='someValuesFrom']/*@*[local-name()=
()='resource']" chebi_lite.owl | cut -d" -f2 | .\getlabels.sh
chebi_lite.owl
```

mutagen
central nervous system stimulant
psychotropic drug
diuretic
xenobiotic
ryanodine receptor modulator
EC 3.1.4.* (phosphoric diester hydrolase) inhibitor
EC 2.7.11.1 (non-specific serine/threonine protein kinase)
inhibitor
adenosine A2A receptor antagonist
adjuvant
food additive
ryanodine receptor agonist
adenosine receptor antagonist
mouse metabolite
plant metabolite
fungal metabolite
environmental contaminant
human blood serum metabolite
Chain invocations of `getparents.sh` until no matches (root)
avoid cyclic relations (infinite loop)
consider only parent relations
Grandparents

Parents of parents also generalizations

Grandparents of *malignant hyperthermia*:

$$\texttt{echo 'malignant hyperthermia' | ./geturi.sh doid.owl | ./getparents.sh doid.owl | .}$$

http://purl.obolibrary.org/obo/DOID_0050739
http://purl.obolibrary.org/obo/DOID_0080000
Their labels:

```
$ echo 'malignant hyperthermia' | ./geturi.sh doid.owl | ./
  getparents.sh doid.owl | ./getparents.sh doid.owl | ./getlabels
  .sh doid.owl

autosomal genetic disease
muscular disease
```
**Root class**

Not have any parent

- *disease* and *chemical entity*
- highly generic terms

**Check root class:**

```
$ echo 'disease' | ./geturi.sh doid.owl | ./getparents.sh doid.owl
$ echo 'chemical entity' | ./geturi.sh chebi_lite.owl | ./getparents.sh chebi_lite.owl
```

**Warning confirming root class:**

XPath set is empty
Script `getancestors.sh`:

1. OWLFILE=$1
2. CLASSES=$(cat -)
3. 
4. PARENTS=$(echo "$CLASSES" | ./getparents.sh $OWLFILE | sort -u)
5. echo "${PARENTS}"
6. echo "${PARENTS}" | ./getancestors.sh $OWLFILE

List of URIs as standard input

invokes `getparents.sh` recursively

until reaches root class
Standard input in variable \texttt{CLASSES} to use twice:
  check input is empty (line 3)
  get parents classes (line 4).

Input empty then script ends
  base case of the recursion
  otherwise run indefinitely

Output in variable \texttt{PARENTS} to use twice
  output these direct parents (line 5)
  get ancestors of parents (line 6)
Invoking `getancestors.sh` inside `getancestors.sh`
defines recursion step
at some time reach classes without parents (root classes)
then script ends

echo of variables CLASSES and PARENTS
inside commas so newline chars preserved
Iteration

Recursion frequently computational expensive
replace recursion with iteration
and explaining iteration
out of scope of this book
Nevertheless, script alternative:

```
1 # iteration
2 OWLFILE=$1
3 CLASSES=$(cat -)
4 ANCESTORS=""
5 while [[ ! -z "$CLASSES" ]]
6 do
7    PARENTS=$(echo "$CLASSES" | ./getparents.sh $OWLFILE | sort -u)
8    ANCESTORS="$ANCESTORS\n$PARENTS"
9    CLASSES=$PARENTS
10 done
11 echo -e "$ANCESTORS"
```

while implements iteration
repeating a set of commands (lines 6-8)
while a condition is satisfied (line 4)
Test with *malignant hyperthermia*:

```bash
$ echo 'http://purl.obolibrary.org/obo/DOID_8545' | ./getancestors.sh doid.owl

http://purl.obolibrary.org/obo/DOID_0050736
http://purl.obolibrary.org/obo/DOID_66
http://purl.obolibrary.org/obo/DOID_0050739
http://purl.obolibrary.org/obo/DOID_0080000
http://purl.obolibrary.org/obo/DOID_0050177
http://purl.obolibrary.org/obo/DOID_17
http://purl.obolibrary.org/obo/DOID_630
http://purl.obolibrary.org/obo/DOID_7
http://purl.obolibrary.org/obo/DOID_4
```
Warning when reaches root class:

XPath set is empty

Redirect warnings:

```bash
$ echo 'malignant hyperthermia' | ./geturi.sh doid.owl | ./
   getancestors.sh doid.owl 2>/dev/null | ./getlabels.sh doid.owl
```
Ancestors of *malignant hyperthermia*:

- autosomal dominant disease
- muscle tissue disease
- autosomal genetic disease
- muscular disease
- monogenic disease
- musculoskeletal system disease
- genetic disease
- disease of anatomical entity
- disease

First two ancestors direct parents

last one the root class.

prints the parents before invoking itself
Same with *caffeine*:

```bash
$ echo 'caffeine' | ./geturi.sh chebi_lite.owl | ./getancestors.sh chebi_lite.owl | ./getlabels.sh chebi_lite.owl | sort -u
```

Repeated classes

- using different branches
- add `sort -u`
Ancestors of *caffeine*:

- alkaloid
- aromatic compound
- bicyclic compound
- carbon group molecular entity
- chemical entity
- cyclic compound
- heteroarene
- heterobicyclic compound
- heterocyclic compound
- heteroorganic entity
- heteropolycyclic compound
- imidazopyrimidine
- main group molecular entity
- methylxanthine
- molecular entity
- molecule
- nitrogen molecular entity
- organic aromatic compound
- organic cyclic compound
- organic heterobicyclic compound
organic heterocyclic compound
organic heteropolycyclic compound
organic molecular entity
organic molecule
organonitrogen compound
organonitrogen heterocyclic compound
p-block molecular entity
pnictogen molecular entity
polyatomic entity
polycyclic compound
purine alkaloid
purines
trimethylxanthine
My Lexicon

Labels and related classes from ontology

Create `do_8545_lexicon.txt`:

```
$ echo 'malignant hyperthermia' | ./geturi.sh doid.owl | ./
getlabels.sh doid.owl > do_8545_lexicon.txt
```

Lexicon for *malignant hyperthermia*

with all its labels
Add to lexicon:

```
$ echo 'malignant hyperthermia' | ./geturi.sh doid.owl | ./
    getancestors.sh doid.owl | ./getlabels.sh doid.owl >> 
    do_8545_lexicon.txt
```

>> and not >

append lines to file
Check contents:

$ cat do_8545_lexicon.txt | sort -u

anesthesia related hyperthermia
autosomal dominant disease
autosomal genetic disease
disease
disease of anatomical entity
genetic disease
malignant hyperpyrexia due to anesthesia
malignant hyperthermia
monogenic disease
muscle tissue disease
muscular disease
musculoskeletal system disease
Same for *caffeine* in `chebi_27732_lexicon.txt`:

```bash
$ echo 'caffeine' | ./geturi.sh chebi_lite.owl | ./getlabels.sh chebi_lite.owl > chebi_27732_lexicon.txt
```

```bash
$ echo 'caffeine' | ./geturi.sh chebi_lite.owl | ./getancestors.sh chebi_lite.owl | ./getlabels.sh chebi_lite.owl >> chebi_27732_lexicon.txt
```

Check contents:

```bash
$ cat chebi_27732_lexicon.txt | sort -u
```

- alkaloid
- aromatic compound
- bicyclic compound
- caffeine
- ... polycyclic compound
- purine alkaloid
- purines
- trimethylxanthine

This lexicon is much larger.
Merging labels

Merging two lexicons in `lexicon.txt`:

```sh
$ cat do_8545_lexicon.txt chebi_27732_lexicon.txt | sort -u > lexicon.txt
```

Recognize any mention in `chebi_27732_sentences.txt`:

```sh
$ grep -w -i -F -f lexicon.txt chebi_27732_sentences.txt
```

-F option

our lexicon is list of fixed strings
not includes regular expressions.
Some results not include direct mention to caffeine or malignant hyperthermia

Example molecule ancestor of caffeine:

The remainder of the molecule is hydrophilic and presumably constitutes the cytoplasmic domain of the protein.

Example disease ancestor of malignant hyperthermia:

Our data suggest that divergent activity profiles may cause varied disease phenotypes by specific mutations.
Use the lexicon with `getentities.sh`:

replace `-E` by the `-F` option

```
$ ./getentities.sh lexicon.txt < chebi_27732_sentences.txt
```
Ancestors matched

Ancestors being matched:
$ grep -o -w -F -f lexicon.txt chebi_27732_sentences.txt | sort -u
  caffeine
disease
malignant hyperthermia
molecule

Text limited and using official labels
  missing acronyms and simple variations (plural)
solution use a stemmer
all ancestors besides subsumption
add some regular expressions
Recognizing any disease represented in ontology in our sentences related to caffeine

Get all labels without restricting to any URI:

```
$ xmllint --xpath "//*[local-name()='Class']/*[local-name()='hasExactSynonym' or local-name()='hasRelatedSynonym' or local-name()='label']/text()" doid.owl
```
Script `getalllabels.sh`:

```bash
1  OWLFILE=$1
2  xmllint --xpath "//*[local-name()='Class']/*[local-name()='hasExactSynonym' or local-name()='hasRelatedSynonym' or local-name()='label']/text()" $OWLFILE | \
3  sort -u
```
Execute:

$ ./getall11labels.sh doid.owl

11-beta-hydroxysteroid dehydrogenase deficiency type 2
11p11.2 deletion
11p partial monosomy syndrome
...
Zoophilia
Zoophobia
zygomycosis
Redirect to *diseases.txt*:

```
$ ./getallllabels.sh doid.owl > diseases.txt
```

Check how many labels:

```
$ wc -l diseases.txt
```

More than 34 thousand labels
Recognize lexicon entries:

$ grep -n -w -E -f diseases.txt chebi_27732_sentences.txt

grep: Unmatched ) or \)

Error because lexicon contains special characters
also used by regular expressions (parentheses)
Replace $-E$ by $-F$:

$\$ grep -n -o -w -F -f diseases.txt chebi_27732_sentences.txt

1:malignant hyperthermia
2:malignant hyperthermia
9:central core disease
10:disease
10:myopathy
...
1092:malignant hyperthermia
1092:central core disease
1103:malignant hyperthermia
1104:malignant hyperthermia
1106:central core disease
1106:myopathy
Problematic entries

Expressions enclosed by parentheses or brackets:
- Post measles encephalitis (disorder)
- Glaucomatous atrophy [cupping] of optic disc

Separation characters (commas or colons)
to represent a specialization
- Tapeworm infection: intestinal taenia solum
- Tapeworm infection: pork
- Pemphigus, Benign Familial
- ATR, nondeletion type

Comma also part of term:
- 46,XY DSD due to LHB deficiency
&amp; to represent ampersand:
Gonococcal synovitis &/or tenosynovitis

But alternatives already included:
Gonococcal synovitis and tenosynovitis
Gonococcal synovitis or tenosynovitis

Not trivial to devise rules
that fully solve these issues
will be exceptions to any rule
Special characters frequency

Check the impact:

```bash
$ grep -c -F '(' diseases.txt
$ grep -c -F ',' diseases.txt
$ grep -c -F '[' diseases.txt
$ grep -c -F ':' diseases.txt
$ grep -c -F '&amp;' diseases.txt
```

Parentheses and commas most frequent
more than one thousand entries
Completeness

Check presence of ATR acronym **alpha thalassemia-X-linked intellectual disability syndrome**

```
$ grep -E '^ATR' diseases.txt

ATR-16 syndrome
ATR, nondeletion type
ATR syndrome, deletion type
ATR syndrome linked to chromosome 16
ATR-X syndrome
```

A single ATR mention will not be recognized:

```
$ echo 'The ATR syndrome is an alpha thalassemia that has material basis in mutation in the ATRX gene on Xq21' | grep -w 'ATR'
```
Removing special characters

Remove parentheses and brackets:

$ tr -d '[()]{}' < diseases.txt

Miss shorter labels such as *Post measles encephalitis*,
but at least will recognize:

$ tr -d '[()]{}' < diseases.txt | grep 'Post measles encephalitis disorder'

Alternative create multiple entries in the lexicon
or transform the labels in regular expressions
Removing extra terms

Remove text after separation char:

$ \text{tr -d '[]{}' } < \text{diseases.txt} | \text{sed -E 's/[,:;] .*$//'}$

Enforces a space after the separation char

avoids: 46,XY DSD due to LHB deficiency

Recognize both ATR and ATR syndrome:

$ \text{tr -d '[]{}' } < \text{diseases.txt} | \text{sed -E 's/[,:;] .*$//'} | \text{grep -E '^ATR'}$
Removing extra spaces

Remove leading or trailing spaces:

```
$ tr -d '[]{}()' < diseases.txt | sed -E 's/[];,.; .*$//; s/^ *///; s/ */$//' 
```

More replacement expressions to sed

separated by semicolon
Update `getalllabels.sh`:

```
1  OWLFFILE=$1
2  xmlmlint --xpath "//*[local-name()='Class']/*[local-name()='hasExactSynonym' or local-name()='hasRelatedSynonym' or local-name()='label']/text()" $OWLFILE | \
3  tr -d '[]{}' | \
4  sed -E 's/[,:;] .*$//; s/^ */;/ s/ */$// | sort -u
```

Generate fixed lexicon:

```
$ ./getalllabels.sh doid.owl > diseases.txt
```
Check number of entries:

$ wc -l diseases.txt

More than 13 thousand labels
less because fixes made duplicate entries
**Disease recognition**

**Recognize entries:**

```
$ grep -n -o -w -F -f diseases.txt chebi_27732_sentences.txt
```

**Labels recognized:**

```
$ grep -o -w -F -f diseases.txt chebi_27732_sentences.txt | sort -u
```

```
47
Andersen-Tawil syndrome
arrhythmogenic right ventricular cardiomyopathy
...
scoliosis
syndrome
T cell
```

47 due to label 47, XXY:
**Performance**

*grep* quite efficient
but large lexicons and texts may give performing issues
execution time proportional to lexicon size
each entry an independent pattern to match

**Inverted Recognition**
uses words of input text as patterns
matched against lexicon
input text smaller than lexicon
*grep* fewer patterns to match
applied to ChEBI 100 times faster
**Case insensitive**

Use the `-i` option:

```bash
$ grep -o -w -F -i -f diseases.txt chebi_27732_sentences.txt | sort -u | wc -l
```

66 labels being recognized.

Check new labels recognized:

```bash
$ grep -o -w -F -i -f diseases.txt chebi_27732_sentences.txt | sort -u > diseases_recognized_ignorecase.txt
$ grep -o -w -F -f diseases.txt chebi_27732_sentences.txt | sort -u > diseases_recognized.txt
$ grep -v -F -f diseases_recognized.txt > diseases_recognized_ignorecase.txt
```
all
All
Arrhythmogenic right ventricular dysplasia
can
Catecholaminergic polymorphic ventricular tachycardia
Central Core Disease
defect
Disease
dyskinesia
face
fever
hypotonia
Malignant hyperthermia
Malignant Hyperthermia
March
ORF
total
Discard case variations with `-f`:

```
$ grep -o -w -F -i -f diseases.txt chebi_27732_sentences.txt | sort
  -u -f | wc -l
```

57 different labels
Correct matches

Some only recognized by case insensitive match
   *dyskinesia*

Lexicon not include lowercase case:

   `$ grep -i -E '^dyskinesia$' diseases.txt`

Lexicon only name with first character in uppercase:
   Dyskinesia
Incorrect matches

Case insensitive match create other problems

CAN acronym of
Crouzon syndrome-acanthosis nigricans syndrome:

$ echo 'CAN' | ./geturi.sh doid.owl | ./getlabels.sh doid.owl

Check how many times CAN is recognized:

$ grep -n -o -w -i -F -f diseases.txt chebi_27732_sentences.txt | grep -i ':CAN' | wc -l

22 times
Which type of matches:

```
$ grep -o -w -i -F -f diseases.txt chebi_27732_sentences.txt | grep
         -i -E '^CAN$' | sort -u
```

Incorrect mentions:

- `can`

22 mismatches by case insensitive match
Entity Linking

What recognized labels represent

Find what $AD2$ represents:

$\text{echo 'AD2' | ./geturi.sh doid.ow}$

Only one URI:

http://purl.obolibrary.org/obo/DOID_0110035
Retrieve labels:

```
$ echo 'http://purl.obolibrary.org/obo/DOID_0110035' | ./getlabels.sh doid.owl
```

Clearly *Alzheimer disease*:

- AD2
- Alzheimer disease 2, late onset
- Alzheimer disease associated with APOE4
- Alzheimer disease-2
- Alzheimer's disease 2
Modified labels

Labels modified by previous fixes:

$ echo 'ATR' | ./geturi.sh doid.owl

XPath set is empty

Solution keep track of the original label
Classes acronym KOS may represent:

$ echo 'KOS' | ./geturi.sh doid.owl

http://purl.obolibrary.org/obo/DOID_0111456
http://purl.obolibrary.org/obo/DOID_0111712

Two distinct diseases:

*Kaufman oculocerebrofacial syndrome* (DOID:0111456)
*Kagami-Ogata syndrome* (DOID:0111712)
Alternative labels:

$ echo 'http://purl.obolibrary.org/obo/DOID_0111456' | ./getlabels.
    sh doid.owl

$ echo 'http://purl.obolibrary.org/obo/DOID_0111712' | ./getlabels.
    sh doid.owl

Both containing **KOS** as expected:

KOS
blepharophimosis ptosis intellectual disability syndrome
oculocerebrofacial syndrome, Kaufman type
Kaufman oculocerebrofacial syndrome

KOS
Kagami-Ogata syndrome
Surrounding entities

Select class closer in meaning to other classes in surrounding text

Assuming entities in same text semantically related
Example:

KOS is a syndromic intellectual disability

Identify the diseases:

$ echo 'KOS is a syndromic intellectual disability' | grep -o -w -f diseases.txt

  -f diseases.txt

KOS
syndromic intellectual disability
Find URIs:

```bash
$ echo 'KOS is a syndromic intellectual disability' | grep -o -w -F»
   -f diseases.txt | ./geturi.sh doid.owl

http://purl.obolibrary.org/obo/DOID_0111456
http://purl.obolibrary.org/obo/DOID_0111712
http://purl.obolibrary.org/obo/DOID_0050888
```

**Syndromic intellectual disability** (DOID:0050888)
Semantic similarity

Solve ambiguity problems
quantify how close two classes are
in terms of semantics
encoded in a given ontology

to calculate semantic similarity between:

* Kaufman oculocerebrofacial syndrome* (DOID:0111456)
* Syndromic intellectual disability* (DOID:0050888)

and

* Kagami-Ogata syndrome* (DOID:0111712)
* Syndromic intellectual disability* (DOID:0050888)
Semantic similarity between *Kaufman oculocerebrofacial syndrome* (DOID:0111456) and *Syndromic intellectual disability* (DOID:0050888)
Semantic similarity between *Kagami-Ogata syndrome* (DOID:0111712) and *Syndromic intellectual disability* (DOID:0050888)
Measures

DiShIn provides three measures
  Resnik, Lin and Jiang-Conrath
last two values between 0 and 1,
Jiang-Conrath distance converted similarity
*Syndromic intellectual disability* more similar to *Kaufman oculocerebrofacial syndrome* than to *Kagami-Ogata syndrome*

Semantic similarity can identify *Kaufman oculocerebrofacial syndrome* correct linked entity for *KOS* in this text
DiShIn installation

Execute DiShIn as a command line
need to install python (or python3)
and SQLite

Download minimalist DiShIn and latest database:

$ curl -O http://labs.rd.ciencias.ulisboa.pt/dishin/dishin.py
$ curl -O http://labs.rd.ciencias.ulisboa.pt/dishin/ssm.py
$ curl -O http://labs.rd.ciencias.ulisboa.pt/dishin/doid202104.db.gz
$ gunzip -N doid202104.db.gz

Full version:

https://github.com/lasigeBioTM/DiShIn
**DiShIn execution**

Semantic similarity between:

*Kauffman oculocerebrofacial syndrome* (DOID:0111456)

*Syndromic intellectual disability* (DOID:0050888)

and

*Kagami-Ogata syndrome* (DOID:0111712)

*Syndromic intellectual disability* (DOID:0050888)

Execute:

$ python dishin.py doid.db DOID_0111456 DOID_0050888

$ python dishin.py doid.db DOID_0111712 DOID_0050888
Semantic similarity between *Kaufman oculocerebrofacial syndrome* (DOID:0111456) and *Syndromic intellectual disability* (DOID:0050888)

<table>
<thead>
<tr>
<th>Method</th>
<th>Metric</th>
<th>Intrinsic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resnik</td>
<td>DiShIn</td>
<td>intrinsic</td>
<td>2.64135297194</td>
</tr>
<tr>
<td>Resnik</td>
<td>MICA</td>
<td>intrinsic</td>
<td>5.28270594387</td>
</tr>
<tr>
<td>Lin</td>
<td>DiShIn</td>
<td>intrinsic</td>
<td>0.382691348274</td>
</tr>
<tr>
<td>Lin</td>
<td>MICA</td>
<td>intrinsic</td>
<td>0.765382696547</td>
</tr>
<tr>
<td>JC</td>
<td>DiShIn</td>
<td>intrinsic</td>
<td>0.105026743844</td>
</tr>
<tr>
<td>JC</td>
<td>MICA</td>
<td>intrinsic</td>
<td>0.235922590328</td>
</tr>
</tbody>
</table>

Semantic similarity between *Kagami-Ogata syndrome* (DOID:0111712) and *Syndromic intellectual disability* (DOID:0050888)

<table>
<thead>
<tr>
<th>Method</th>
<th>Metric</th>
<th>Intrinsic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resnik</td>
<td>DiShIn</td>
<td>intrinsic</td>
<td>0.0</td>
</tr>
<tr>
<td>Resnik</td>
<td>MICA</td>
<td>intrinsic</td>
<td>0.0</td>
</tr>
<tr>
<td>Lin</td>
<td>DiShIn</td>
<td>intrinsic</td>
<td>0.0</td>
</tr>
<tr>
<td>Lin</td>
<td>MICA</td>
<td>intrinsic</td>
<td>-0.0</td>
</tr>
<tr>
<td>JC</td>
<td>DiShIn</td>
<td>intrinsic</td>
<td>0.0675488987867</td>
</tr>
<tr>
<td>JC</td>
<td>MICA</td>
<td>intrinsic</td>
<td>0.0675488987867</td>
</tr>
</tbody>
</table>

Learning python and SQL
out of scope of this book
but quite simple to execute
Large lexicons

Online tool MER
- a shell script
- easily executed as a command line
- efficiently recognize and link entities
- using large lexicons
MER installation

Install it locally:

```bash
$ unzip master.zip
$ mv MER-master MER
```

Copy Human Disease Ontology:

```bash
$ cp doid.owl MER/data/
$ cd MER
```
Lexicon files

Create lexicon:

```bash
$ (cd data; ../produce_data_files.sh doid.owl)
```

Check the contents:

```bash
$ tail data/doid_*
```

```plaintext
==> data/doid_links.tsv <==
  ziziphus mauritiana fruit allergy http://purl.obolibrary.org/obo/DOID_0060507
  zlotogora-ogur syndrome http://purl.obolibrary.org/obo/DOID_0080400
  zlotogora-zilberman-tenenbaum syndrome http://purl.obolibrary.org/obo/DOID_0060773
  zollinger-ellison syndrome http://purl.obolibrary.org/obo/DOID_0050782
  zoophilia http://purl.obolibrary.org/obo/DOID_9336
  zoophobia http://purl.obolibrary.org/obo/DOID_600
```
zunich-kaye syndrome http://purl.obolibrary.org/obo/DOID_0112152
zunich neuroectodermal syndrome http://purl.obolibrary.org/obo/DOID_0112152
zygodactyly 1 http://purl.obolibrary.org/obo/DOID_0111820
zygomyconsis http://purl.obolibrary.org/obo/DOID_8485

cmd> data/doid_word1.txt
xpid
xpv
xrn
xscid
yaba
yaws
zaspopathy
zoophilia
zoophobia
zygomyconsis

ncmd> data/doid_word2.txt
zellweger syndrome
zemuron allergy
semantic processing

Semantic Processing

Large lexicons

zika fever
zinacef allergy
zinssser.cole.engman syndrome
zlotogora.ogur syndrome
zlotogora.zilberman.tenenbaum syndrome
zollinger.ellison syndrome
zunich.kaye syndrome
zygodactyly 1

==> data/doid_words2.txt <==
y.linked monogenic
y.linked sertoli
y.linked spermatogenic
yolk sac
young adult.onset
zeta.associated.protein 70
zika virus
zikv congenital
ziziphus mauritiana
zunich neuroectodermal
yolk sac tumour
yolk sac tumour of the cns
young adult.onset dhmn
young adult.onset distal hereditary motor neuropathy
zeta.associated.protein 70 deficiency
zika virus congenital syndrome
zika virus disease
zikv congenital infection
ziziphus mauritiana fruit allergy
zunich neuroectodermal syndrome
MER execution

Execute MER:

```bash
$ cat ../chebi_27732_sentences.txt | tr -d "'" | xargs -I {} ./
  get_entities.sh '{}
```

Removes single quotes special characters to `xargs`.

`get_entities.sh` script inside MER folder not the one created before
Large number of matches:

89 111 malignant hyperthermia http://purl.obolibrary.org/obo/DOID_8545
74 96 malignant hyperthermia http://purl.obolibrary.org/obo/DOID_8545
157 164 disease http://purl.obolibrary.org/obo/DOID_4
144 164 central core disease http://purl.obolibrary.org/obo/DOID_3529
13 20 disease http://purl.obolibrary.org/obo/DOID_4
47 55 myopathy http://purl.obolibrary.org/obo/DOID_423

... 

First two numbers represent the start and end position of match followed by label and its URI
Create *diseases_recognized.tsv*:

```
$ cat ../chebi_27732_sentences.txt | tr -d "" | xargs -I {} ./
    get_entities.sh '{}'' doid > ../diseases_recognized.tsv
```

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>89</td>
<td>malignant hyperthermia</td>
<td><a href="http://purl.obolibrary.org/obo/DOID_8545">http://purl.obolibrary.org/obo/DOID_8545</a></td>
</tr>
<tr>
<td>2</td>
<td>144</td>
<td>central core disease</td>
<td><a href="http://purl.obolibrary.org/obo/DOID_3529">http://purl.obolibrary.org/obo/DOID_3529</a></td>
</tr>
<tr>
<td>3</td>
<td>13</td>
<td>disease</td>
<td><a href="http://purl.obolibrary.org/obo/DOID_4">http://purl.obolibrary.org/obo/DOID_4</a></td>
</tr>
<tr>
<td>4</td>
<td>47</td>
<td>myopathy</td>
<td><a href="http://purl.obolibrary.org/obo/DOID_423">http://purl.obolibrary.org/obo/DOID_423</a></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>Central core disease</td>
<td><a href="http://purl.obolibrary.org/obo/DOID_3529">http://purl.obolibrary.org/obo/DOID_3529</a></td>
</tr>
<tr>
<td>6</td>
<td>267</td>
<td>disease</td>
<td><a href="http://purl.obolibrary.org/obo/DOID_4">http://purl.obolibrary.org/obo/DOID_4</a></td>
</tr>
<tr>
<td>7</td>
<td>254</td>
<td>central core disease</td>
<td><a href="http://purl.obolibrary.org/obo/DOID_3529">http://purl.obolibrary.org/obo/DOID_3529</a></td>
</tr>
<tr>
<td>8</td>
<td>48</td>
<td>malignant hyperthermia</td>
<td><a href="http://purl.obolibrary.org/obo/DOID_8545">http://purl.obolibrary.org/obo/DOID_8545</a></td>
</tr>
</tbody>
</table>

The *diseases_recognized.tsv* file opened in a spreadsheet application
More: