

Data and Text Processing for Health and Life Sciences

Francisco M. Couto

March 20, 2020



Book website: <http://labs.rd.ciencias.ulisboa.pt/book/>

First Edition: <https://link.springer.com/book/10.1007/978-3-030-13845-5>

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

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can be outdated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

Scientific text

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

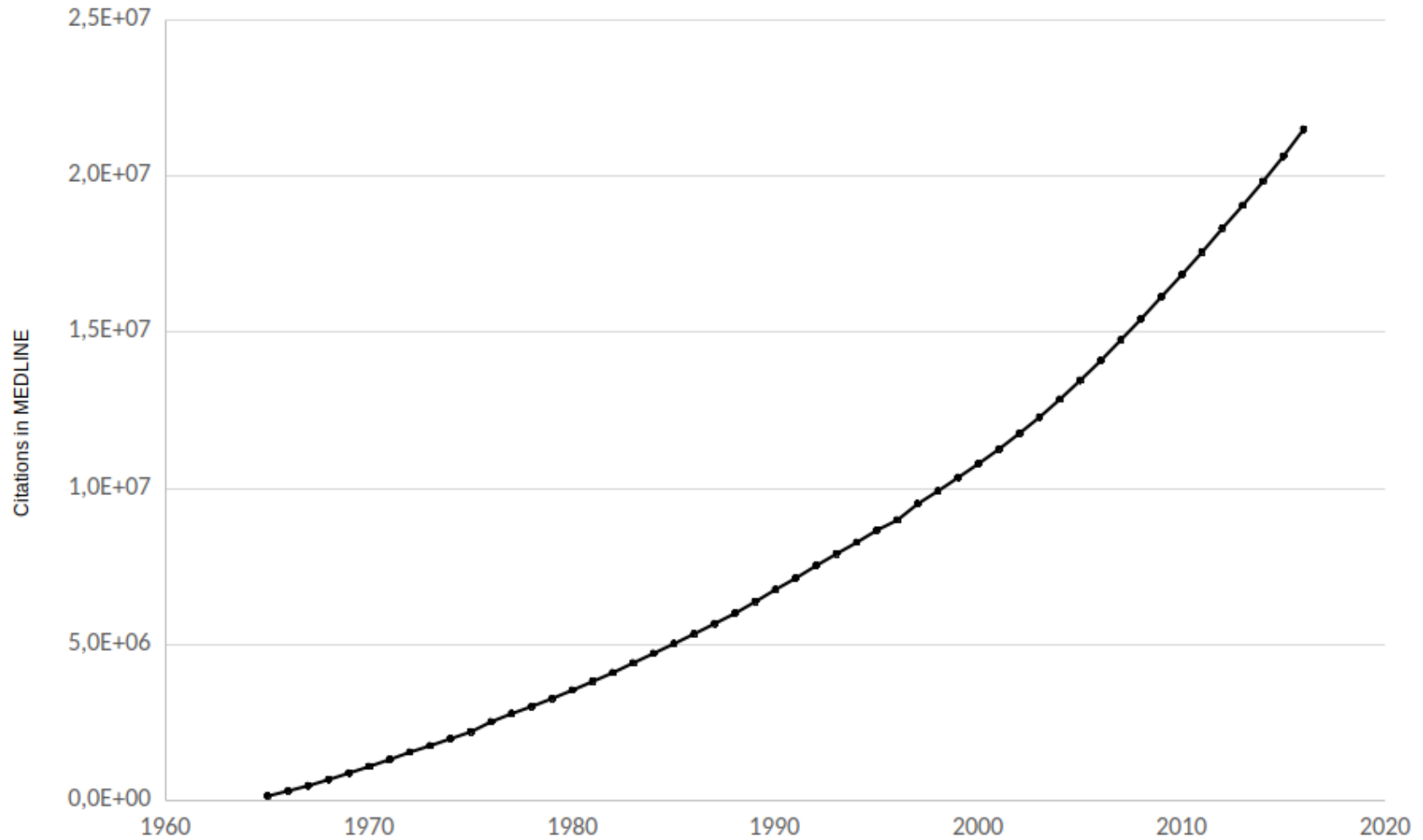
Text is still the preferential means to publish new discoveries and to describe the data that support them

Amount of text

Huge amount of text being published every day

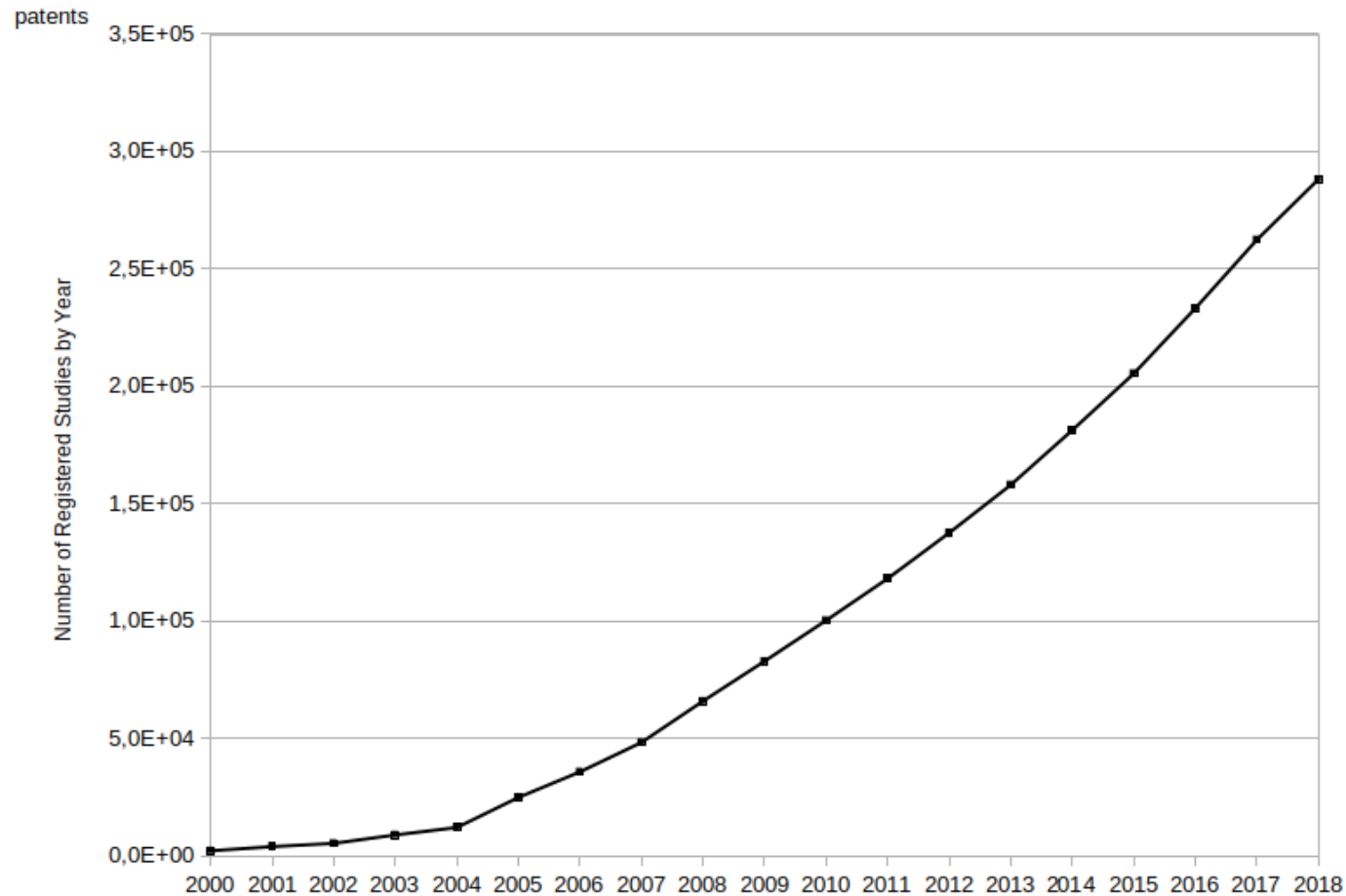
For example, 813,598 citations were added in 2017 to MEDLINE
10 articles per day
more than 222 years to read those articles

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



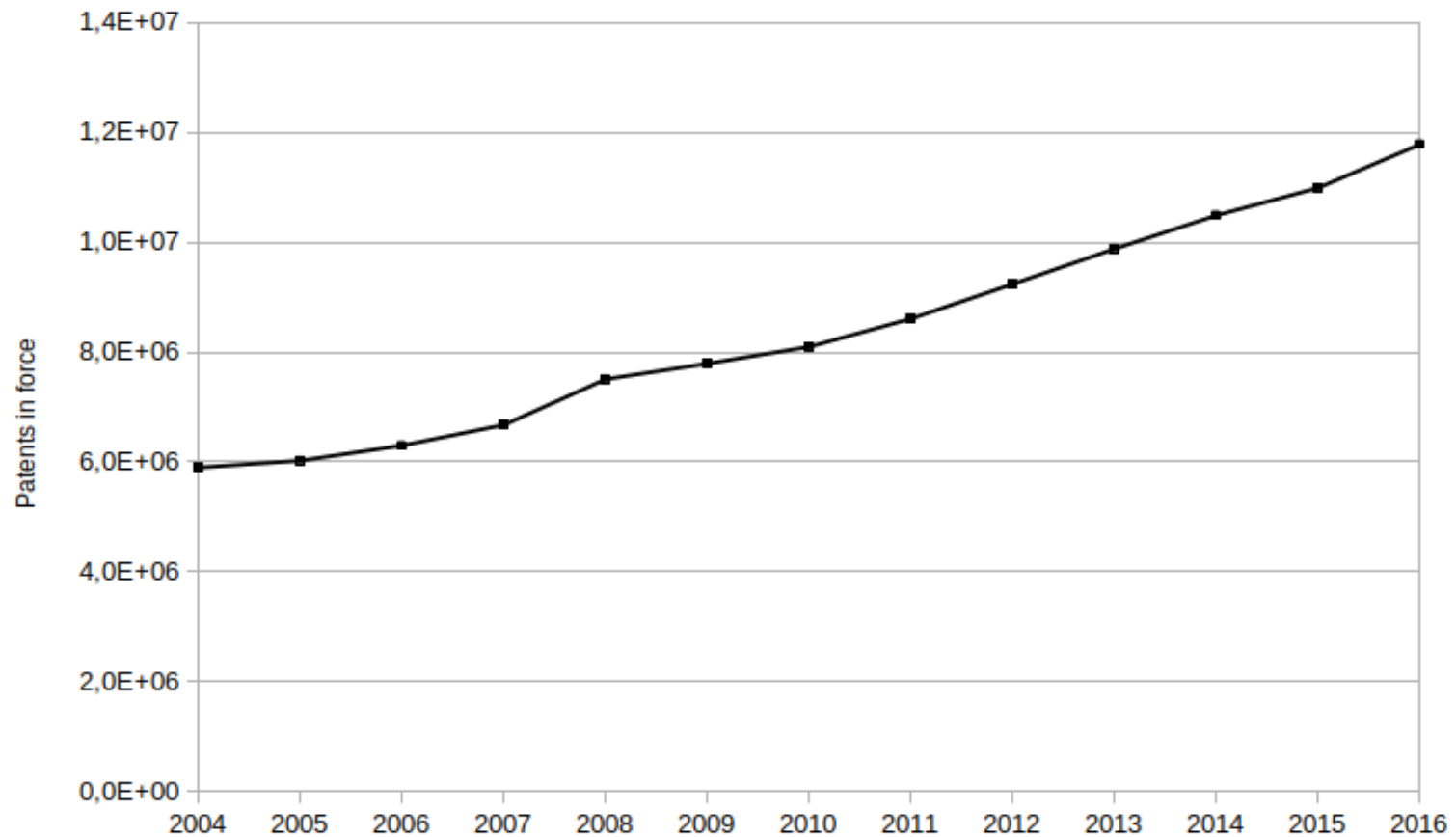
Total number of citations in MEDLINE

(Source: <https://www.nlm.nih.gov/bsd/>)



Total number of registered studies (clinical trials)

(Source: <https://clinicaltrials.gov>)



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 pro-
gramming?**

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
changes on how resources are being distributed
software and hardware updates

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

Open-source solutions may seem a great alternative
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
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

	A	B
1	A	C
2	G	T
3		

Spreadsheet example

CSV file contents:

A, C

G, T

TSV file contents:

A C

G T

XML file contents:

...

```
<Table ss:StyleID="ta1">
```

```
<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>
```

...

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 *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;
 - `gawk`: a tool to manipulate our data and text;
 - `sed`: a tool to edit our data and text;
 - `xargs`: a tool to repeat the same step for multiple data items;
 - `xmllint`: 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

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

How?

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

Search for *caffeine* using the UniProt Citations
select the first two entries,
click on download

In the browser (tabular format):

PubMed ID	Title	Authors/Groups	Abstract/Summary
27702941	Genome-wide association ...		
22333316	Modeling caffeine concentrations ...		

URL used:

```
https://www.uniprot.org/citations/?sort=score&desc=&  
compress=no&query=id:27702941%20OR%20id:22333316&  
format=tab&columns=id
```

Components:

- scheme (`https`)
- the hostname (`www.uniprot.org`)
- the service (`citations`)
- and the data parameters

Good news

- use this link with a command line tool
- automatize the retrieval of the data

Change any value of the parameters (arguments)
the PubMed id 29029291:

```
https://www.uniprot.org/citations/?sort=score&desc=&  
compress=no&query=id:29029291&format=tab&columns=id
```

In the browser:

PubMed ID	Title	Authors/Groups	Abstract/Summary
29029291	Nutrition	Influences	...

Semantics

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.

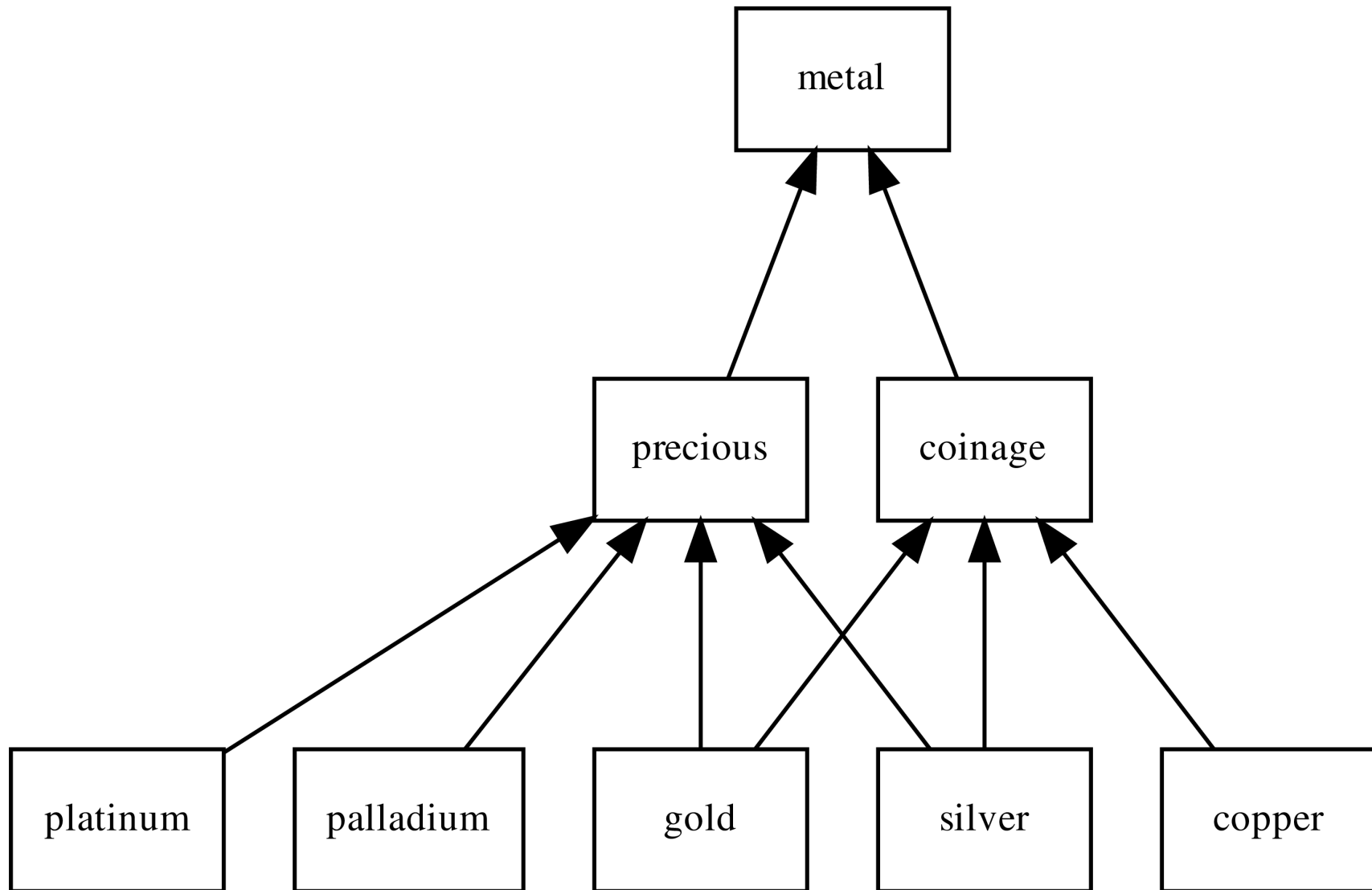
include multiple inheritance

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

Where?

BioPortal (Dec. 2018)

- more than 750 ontologies

- more than 9 million classes (2018)

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

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)
187 ontologies (Dec. 2018)

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

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:

```
<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

URI

The Uniform Resource Identifier (URI)

standard global identifier of classes

class `caffeine` in ChEBI identified by:

`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)

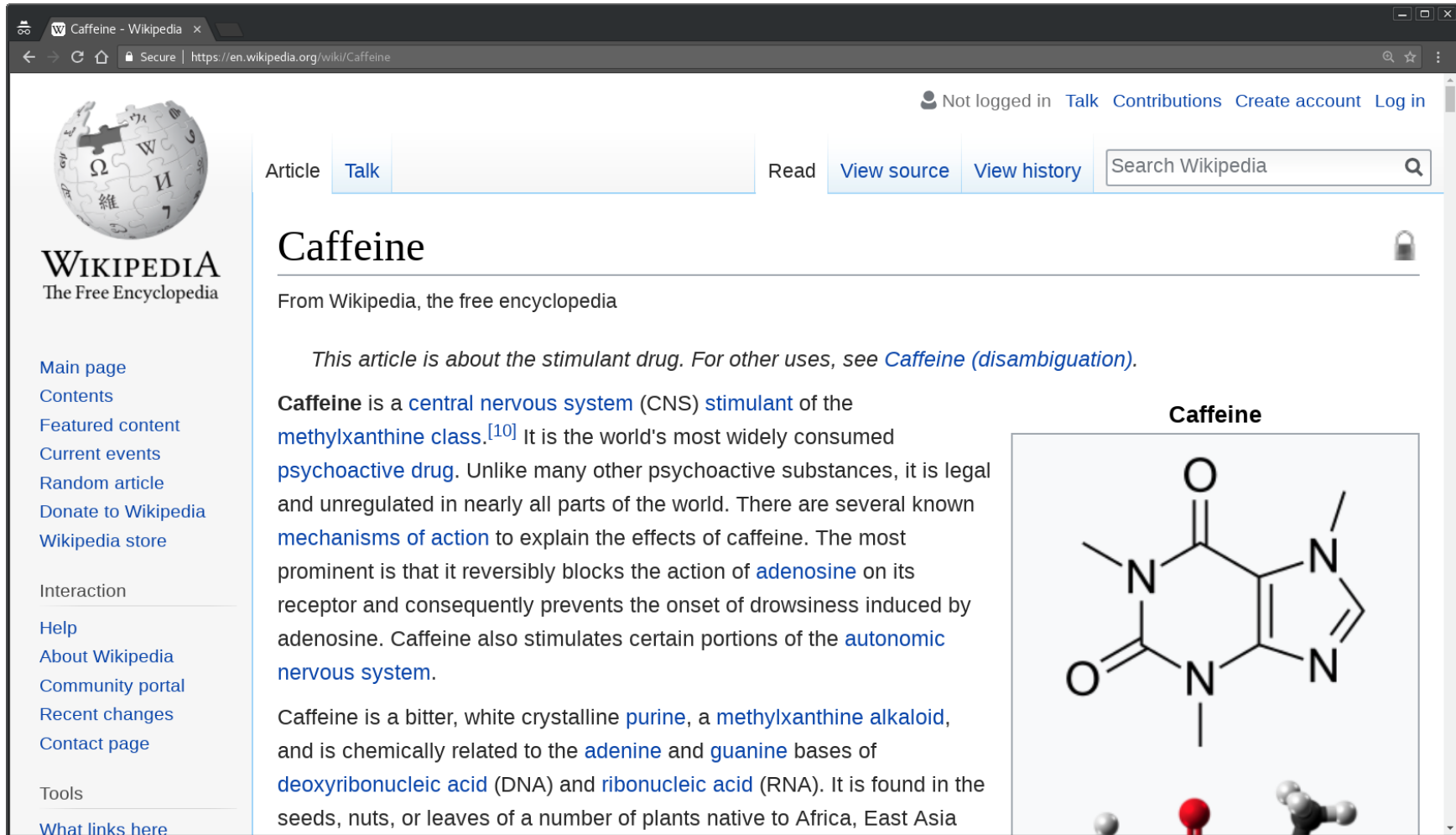
Data Retrieval

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



The image is a screenshot of the Wikipedia page for "Caffeine". The browser window shows the URL "https://en.wikipedia.org/wiki/Caffeine". The page features the Wikipedia logo on the left, a navigation menu with links like "Main page", "Contents", and "Help", and a search bar at the top right. The article title "Caffeine" is prominently displayed, followed by a sub-header "From Wikipedia, the free encyclopedia". A disclaimer states: "This article is about the stimulant drug. For other uses, see [Caffeine \(disambiguation\)](#)." The main text describes caffeine as a "central nervous system (CNS) stimulant" and a "methylxanthine class" compound, noting its widespread consumption and legal status. It details its mechanism of action, specifically its reversible blockade of adenosine receptors. To the right of the text is a chemical structure diagram of caffeine, labeled "Caffeine", showing its purine ring system with two methyl groups and two carbonyl groups. Below the diagram is a small 3D ball-and-stick model of the molecule.

Wikipedia page about caffeine

<div><div>Caffeine - Wikipedia</div><div>Secure https://en.wikipedia.org/wiki/Caffeine</div></div>			
<div><div><div>1.2 Enhancing performance</div><div>1.3 Specific populations</div><div>2 Adverse effects</div><div>2.1 Physical</div><div>2.2 Psychological</div><div>2.3 Reinforcement disorders</div><div>2.4 Risk of other diseases</div><div>3 Overdose</div><div>4 Interactions</div><div>4.1 Alcohol</div><div>4.2 Tobacco</div><div>4.3 Birth control</div><div>4.4 Medications</div><div>5 Pharmacology</div><div>5.1 Pharmacodynamics</div><div>5.2 Pharmacokinetics</div><div>6 Chemistry</div><div>6.1 Synthesis</div><div>6.2 Decaffeination</div><div>6.3 Detection in body fluids</div><div>6.4 Analogs</div><div>6.5 Precipitation of tannins</div><div>7 Natural occurrence</div></div></div>		<div><div>Excretion</div><div>Urine (100%)</div><div>Identifiers</div><div><div>IUPAC name</div><div>[show]</div></div><div><div>CAS Number</div><div>58-08-2</div></div><div><div>PubChem CID</div><div>2519</div></div><div><div>IUPHAR/BPS</div><div>407</div></div><div><div>DrugBank</div><div>DB00201</div></div><div><div>ChemSpider</div><div>2424</div></div><div><div>UNII</div><div>3G6A5W338E</div></div><div><div>KEGG</div><div>D00528</div></div><div><div>ChEBI</div><div>CHEBI:27732</div></div><div><div>ChEMBL</div><div>CHEMBL113</div></div><div><div>PDB ligand</div><div>CFF (PDBe, RCSB PDB)</div></div><div><div>ECHA InfoCard</div><div>100.000.329</div></div><div>Chemical and physical data</div><div><div>Formula</div><div>C₈H₁₀N₄O₂</div></div><div><div>Molar mass</div><div>194.19 g/mol</div></div><div><div>3D model (JSmol)</div><div>Interactive image</div></div><div><div>Density</div><div>1.23 g/cm³</div></div><div><div>Melting point</div><div>235 to 238 °C (455 to 460 °F) (anhydrous)^{[8][9]}</div></div><div><div>SMILES</div><div>[show]</div></div><div><div>InChI</div><div>[show]</div></div></div>	

Identifiers section of the Wikipedia page about caffeine

The screenshot shows the ChEBI (Chemical Entities of Biological Interest) website interface. The browser address bar displays the URL: <https://www.ebi.ac.uk/chebi/searchId.do?chebiId=CHEBI:27732>. The ChEBI logo is prominently displayed at the top left. A search bar is located at the top right, with a search button and a link to 'Advanced' search. Below the search bar, a navigation menu includes links for Home, Advanced Search, Browse, Documentation, Download, Tools, and About ChEBI. The main content area is titled 'CHEBI:27732 - caffeine'. It features a tabbed interface with 'Main' selected, and other tabs for 'ChEBI Ontology', 'Automatic Xrefs', 'Reactions', 'Pathways', and 'Models'. On the left, the chemical structure of caffeine is shown. To the right, a table-like layout provides key information:

ChEBI Name	caffeine
ChEBI ID	CHEBI:27732
Definition	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.
Stars	☆☆☆ This entity has been manually annotated by the ChEBI Team.
Secondary ChEBI IDs	CHEBI:3295, CHEBI:41472, CHEBI:22982
Supplier Information	ZINC000000001084 , eMolecules:493944 , eMolecules:27517656
Download	Molfile XML SDF

At the bottom, there is a link: 'Find compounds which contain this structure'.

ChEBI entry describing caffeine

The screenshot shows the ChEBI (Chemical Entities of Biological Interest) website interface for caffeine (CHEBI:277). The browser address bar shows the URL: <https://www.ebi.ac.uk/chebi/displayAutoXrefs.do?chebiId=CHEBI:27732>. The page has a navigation bar with tabs: Main, ChEBI Ontology, Automatic Xrefs, Reactions, Pathways, and Models. Below the navigation bar, there are expand/collapse controls for 'Expand relevant' and 'Collapse all'. The main content area is divided into two columns. The left column contains 'Protein Sequences' (77 items) and 'Small molecules' (21 items). The right column contains 'Reactions & Pathways' (18 items). Each section lists external references with a 'show all >>' link.

Category	Count
Protein Sequences	77
Small molecules	21
Reactions & Pathways	18

Protein Sequences

- UniProt KB** (77)
UniProt Knowledge Base of protein sequences.
 - [A2AGL3](#)
Ryanodine receptor 3
 - [A4GE69](#)
7-methylxanthosine synthase 1
 - [A4GE70](#)
3,7-dimethylxanthine N-methyltransferase
 - [A6MFK9](#)
Cysteine-rich venom protein
 - [B0LPN4](#)
Ryanodine receptor 2[show all >>](#)
- NMRShiftDB** (1)
NMRShiftDB is a NMR database for organic structures and their nuclear magnetic resonance (nmr) spectra.
 - [10016316](#)

Reactions & Pathways

- BioModels** (2)
Database of Mathematical models of biological interest.
 - [BIOMD0000000241](#)
Shi1993_Caffeine_pressor_tolerance
 - [BIOMD0000000601](#)
Rosas2015 - Caffeine-induced luminal SR calcium changes[show all >>](#)
- BKMS-react** (3)
BKMS-react is an integrated and non-redundant biochemical reaction database containing known enzyme-catalyzed and spontaneous reactions.
 - [882](#)
 - [7965](#)
 - [51266](#)[show all >>](#)
- Rhea** (6)
Rhea is a freely available, manually annotated database of biochemical reactions.

External references related to caffeine

UniProt Automatically Generated Cross-References

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

77 entries found, displaying 1 to 15.

Identifiers	Name	Line Types
A2AGL3	Ryanodine receptor 3	CC - MISCELLANEOUS
A4GE69	7-methylxanthosine synthase 1	CC - FUNCTION
A4GE70	3,7-dimethylxanthine N-methyltransferase	CC - CATALYTIC ACTIVITY; CC - FUNCTION
A6MFK9	Cysteine-rich venom protein	CC - FUNCTION
B0LPN4	Ryanodine receptor 2	CC - MISCELLANEOUS
B7FDI0	Cysteine-rich venom protein	CC - FUNCTION
B7FDI1	Cysteine-rich venom protein	CC - FUNCTION
B8QG00	Hadrucalcin	CC - FUNCTION
D7REY3	Caffeine dehydrogenase subunit alpha	DE; FT; CC - CATALYTIC ACTIVITY; CC - FUNCTION; CC - BIOPHYSICOCHEMICAL PROPERTIES
D7REY4	Caffeine dehydrogenase subunit beta	DE; FT; CC - CATALYTIC ACTIVITY; CC - FUNCTION; CC - BIOPHYSICOCHEMICAL PROPERTIES
D7REY5	Caffeine dehydrogenase subunit gamma	DE; FT; CC - CATALYTIC ACTIVITY; CC - FUNCTION; CC - BIOPHYSICOCHEMICAL PROPERTIES
E9PZQ0	Ryanodine receptor 1	CC - MISCELLANEOUS
E9Q401	Ryanodine receptor 2	CC - MISCELLANEOUS
F0E1K6	Probable methylxanthine N7-demethylase NdmC	CC - FUNCTION
F1LMY4	Ryanodine receptor 1	CC - MISCELLANEOUS

Export options: [CSV](#) | [Excel](#) | [XML](#)

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:

"Identifiers"	"Name"	"Line
"A2AGL3"	"Ryanodine receptor 3"	"CC - MISCELLANEOUS"
"A4GE69"	"7-methylxanthosine synthase 1"	"CC - FUNCTION"
...		

chebi_27732_xrefs_UniProt.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>
...
```

UniProtKB - P21817 (RYR1_HUMAN)

Display

Entry

Publications

Feature viewer

Feature table

Protein | **Ryanodine receptor 1**

Gene | **RYR1**

Organism | *Homo sapiens (Human)*

Status | **Reviewed** - Annotation score: - Experimental evidence at protein levelⁱ

Functionⁱ

Calcium channel that mediates the release of Ca^{2+} from the sarcoplasmic reticulum into the cytoplasm and thereby plays a key role in triggering muscle contraction following depolarization of T-tubules (PubMed:11741831, PubMed:16163667). Repeated very high-level exercise increases the open probability of the channel and leads to Ca^{2+} leaking into the cytoplasm (PubMed:18268335). Can also mediate the release of Ca^{2+} from intracellular stores in neurons, and may thereby promote prolonged Ca^{2+} signaling in the brain. Required for normal embryonic development of muscle fibers and skeletal muscle. Required for normal heart morphogenesis, skin development and ossification during embryogenesis (By similarity).

By similarity 2 Publications 1 Publication

Miscellaneous

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" xmlns:xsi=
  "http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://uniprot.org/uniprot http:
    //www.uniprot.org/support/docs/uniprot.xsd">
<entry dataset="Swiss-Prot" created="1991-05-01"
  modified="2018-06-20" version="210">
<accession>P21817</accession>
...
```


Homo sapiens (Human) protein
interested only in Human Proteins
filter them

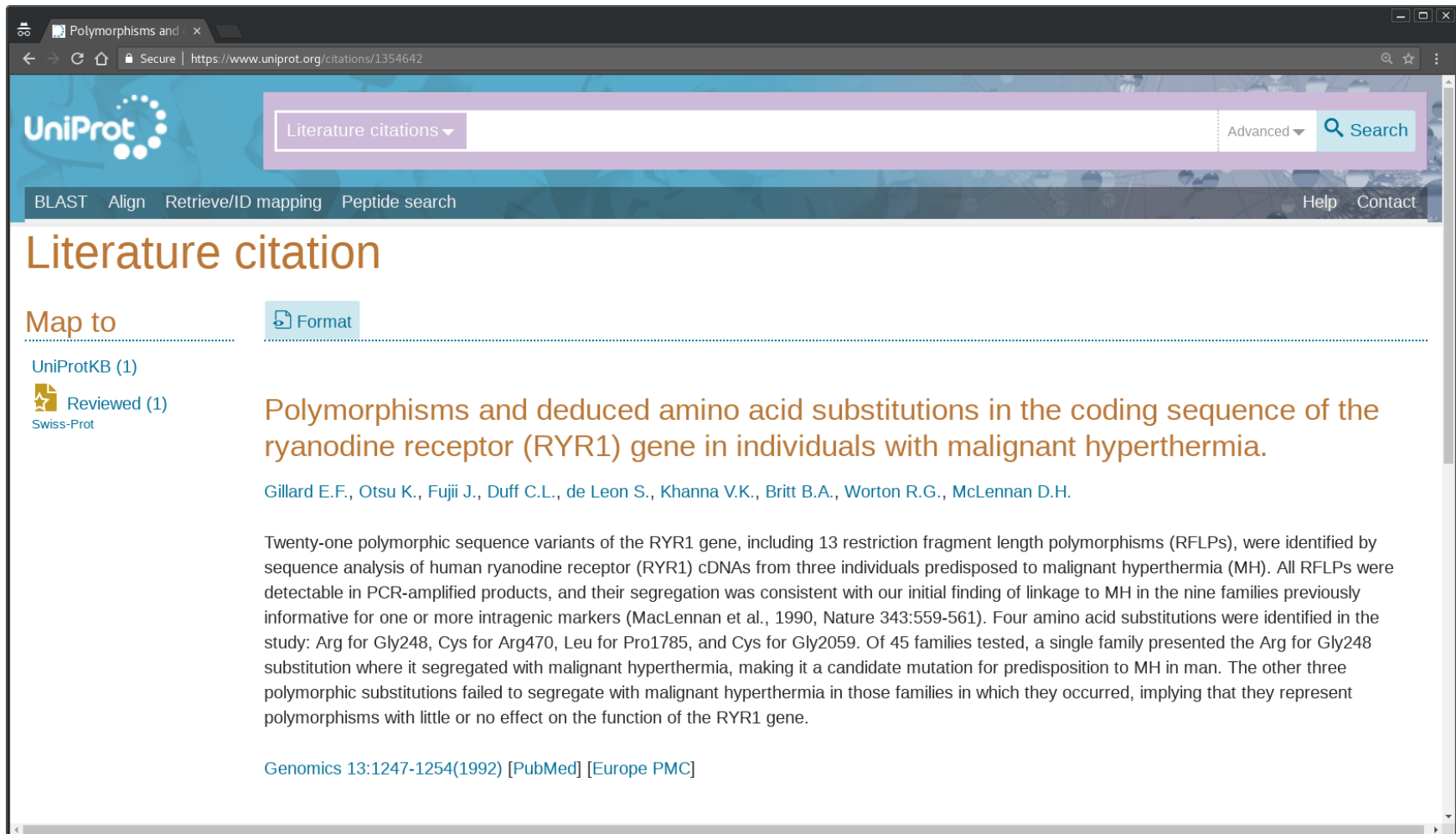
Entry E9PZQ0
Ryanodine receptor 1 protein
but *Mus musculus (Mouse)*

The screenshot shows the UniProt website interface. The browser address bar displays "https://www.uniprot.org/uniprot/P21817/publications". The UniProt logo is in the top left. A search bar with "UniProtKB" and a search button is at the top. Below the search bar, navigation links include "BLAST", "Align", "Retrieve/ID mapping", "Peptide search", "Help", and "Contact". The main heading is "Publications related to P21817 - RYR1_HUMAN". On the right, there is a "Basket" button and a pagination control showing "1 to 25 of 237" and a "Show 25" dropdown. On the left, a sidebar contains links: "Display" (with sub-links "Entry", "Publications", "Feature viewer", "Feature table"), "Filter by", "Source" (with sub-links "UniProtKB/Swiss-Prot (78) (reviewed)", "Computationally mapped (159)"), and "Category". The main content area lists three publications:

- "Molecular cloning of cDNA encoding human and rabbit forms of the Ca²⁺ release channel (ryanodine receptor) of skeletal muscle sarcoplasmic reticulum."**
Zorzato F., Fujii J., Otsu K., Phillips M.S., Green N.M., Lai F.A., Meissner G., MacLennan D.H.
J. Biol. Chem. 265:2244-2256(1990) [PubMed] [Europe PMC] [Abstract]
Cited for: NUCLEOTIDE SEQUENCE [MRNA] (ISOFORM 2), PARTIAL PROTEIN SEQUENCE.
Category: Sequences.
Tissue: Skeletal muscle.
Source: UniProtKB/Swiss-Prot (reviewed).
This publication is cited by 1 and mapped to 9 other entries.
- "Polymorphisms and deduced amino acid substitutions in the coding sequence of the ryanodine receptor (RYR1) gene in individuals with malignant hyperthermia."**
Gillard E.F., Otsu K., Fujii J., Duff C.L., de Leon S., Khanna V.K., Britt B.A., Worton R.G., McLennan D.H.
Genomics 13:1247-1254(1992) [PubMed] [Europe PMC] [Abstract]
Cited for: SEQUENCE REVISION TO 2324; 2840 AND 3380, INVOLVEMENT IN MHS1, VARIANT MHS1 ARG-248, VARIANTS CYS-471; LEU-1787; CYS-2060 AND VAL-2550.
Category: Pathology & Biotech, Sequences.
Tissue: Muscle.
Source: UniProtKB/Swiss-Prot (reviewed).
- "A mutation in the human ryanodine receptor gene associated with central core disease."**

Publications related to Ryanodine receptor 1

Top-left link to publications
click on it
list of publications somehow related to the protein



The screenshot shows a web browser window with the UniProt website. The address bar displays the URL <https://www.uniprot.org/citations/1354642>. The UniProt logo is in the top left. A search bar with a dropdown menu set to 'Literature citations' and a 'Search' button is in the top right. Below the search bar, navigation links include 'BLAST', 'Align', 'Retrieve/ID mapping', 'Peptide search', 'Help', and 'Contact'. The main heading is 'Literature citation'. Under 'Map to', there are links for 'UniProtKB (1)', 'Reviewed (1)', and 'Swiss-Prot'. A 'Format' button is also present. The title of the citation is 'Polymorphisms and deduced amino acid substitutions in the coding sequence of the ryanodine receptor (RYR1) gene in individuals with malignant hyperthermia.' The authors listed are Gillard E.F., Otsu K., Fujii J., Duff C.L., de Leon S., Khanna V.K., Britt B.A., Worton R.G., and McLennan D.H. The abstract text describes the identification of 21 polymorphic sequence variants of the RYR1 gene, including 13 RFLPs, and their segregation with malignant hyperthermia (MH) in nine families. It mentions four amino acid substitutions: Arg for Gly248, Cys for Arg470, Leu for Pro1785, and Cys for Gly2059. The citation is from Genomics 13:1247-1254(1992), with links to PubMed and Europe PMC.

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

Gillard E.F., Otsu K., Fujii J., Duff C.L., de Leon S., Khanna V.K., Britt B.A., Worton R.G., McLennan D.H.

Twenty-one polymorphic sequence variants of the RYR1 gene, including 13 restriction fragment length polymorphisms (RFLPs), were identified by sequence analysis of human ryanodine receptor (RYR1) cDNAs from three individuals predisposed to malignant hyperthermia (MH). All RFLPs were detectable in PCR-amplified products, and their segregation was consistent with our initial finding of linkage to MH in the nine families previously informative for one or more intragenic markers (MacLennan et al., 1990, Nature 343:559-561). Four amino acid substitutions were identified in the study: Arg for Gly248, Cys for Arg470, Leu for Pro1785, and Cys for Gly2059. Of 45 families tested, a single family presented the Arg for Gly248 substitution where it segregated with malignant hyperthermia, making it a candidate mutation for predisposition to MH in man. The other three polymorphic substitutions failed to segregate with malignant hyperthermia in those families in which they occurred, implying that they represent polymorphisms with little or no effect on the function of the RYR1 gene.

[Genomics 13:1247-1254\(1992\)](#) [[PubMed](#)] [[Europe PMC](#)]

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

MER - Minimal Entity Recognizer

Lexicon
DO - Human Disease Ontology

Text
Twenty-one polymorphic sequence variants of the RYR1 gene, including 13 restriction fragment length polymorphisms (RFLPs), were identified by sequence analysis of human ryanodine receptor (RYR1) cDNAs from three individuals predisposed to malignant hyperthermia (MH). All RFLPs were detectable in PCR-amplified products, and their segregation was consistent with our initial finding of linkage to MH in the nine families previously informative for one or more intragenic markers (MacLennan et al., 1990, Nature 343:559-561). Four amino acid substitutions were identified in the study: Arg for Gly248, Cys for Arg470, Leu for Pro1785, and Cys for Gly2059. Of 45 families tested, a single family presented the Arg for Gly248 substitution where it segregated with malignant hyperthermia, making it a candidate mutation for predisposition to MH in man. The other three polymorphic substitutions failed to segregate with malignant hyperthermia in those families in which they occurred, implying that they represent polymorphisms with little or no effect on the function of the RYR1 gene.

Submit

Start	End	Text	Link
239	261	malignant hyperthermia	http://purl.obolibrary.org/obo/DOID_8545
761	783	malignant hyperthermia	http://purl.obolibrary.org/obo/DOID_8545
916	938	malignant hyperthermia	http://purl.obolibrary.org/obo/DOID_8545

Diseases recognized by the online tool MER in an abstract

Mentions any disease

use an online text mining tool

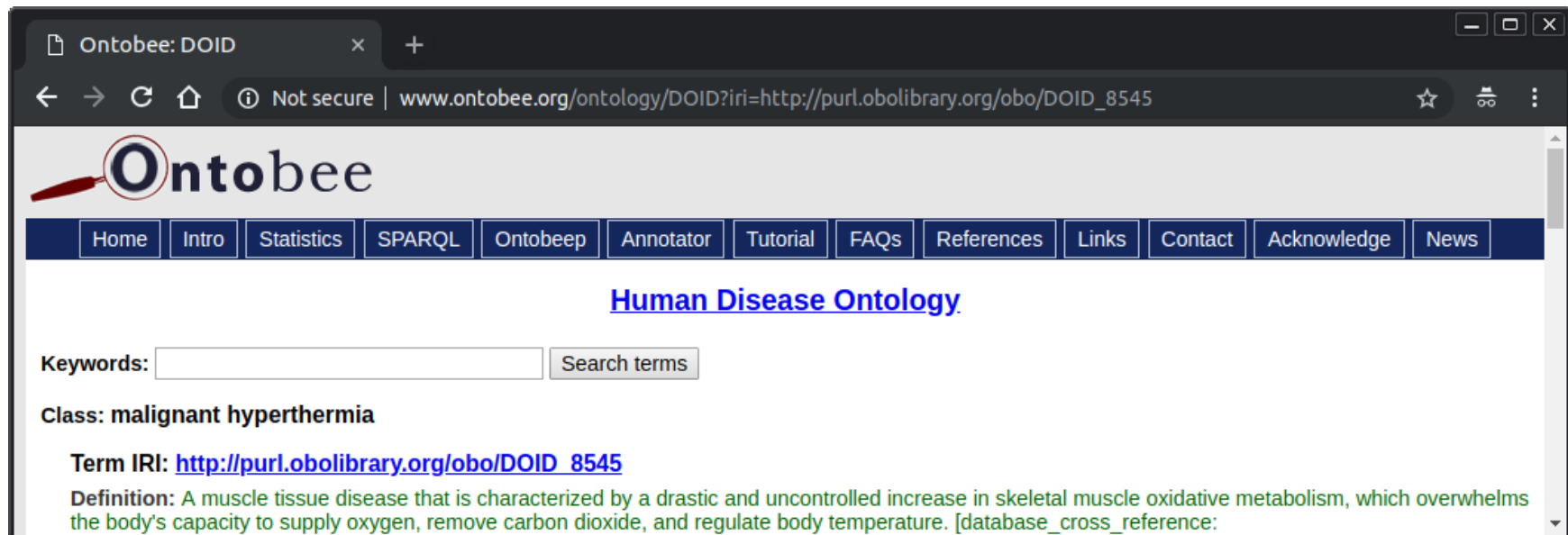
Minimal Named-Entity Recognizer (MER)

<http://labs.rd.ciencias.ulisboa.pt/mer/>.

copy and paste the abstract

select *DO - Human Disease Ontology* as lexicon

Detects three mentions of *malignant hyperthermia*,
link about the disease



Ontobee 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

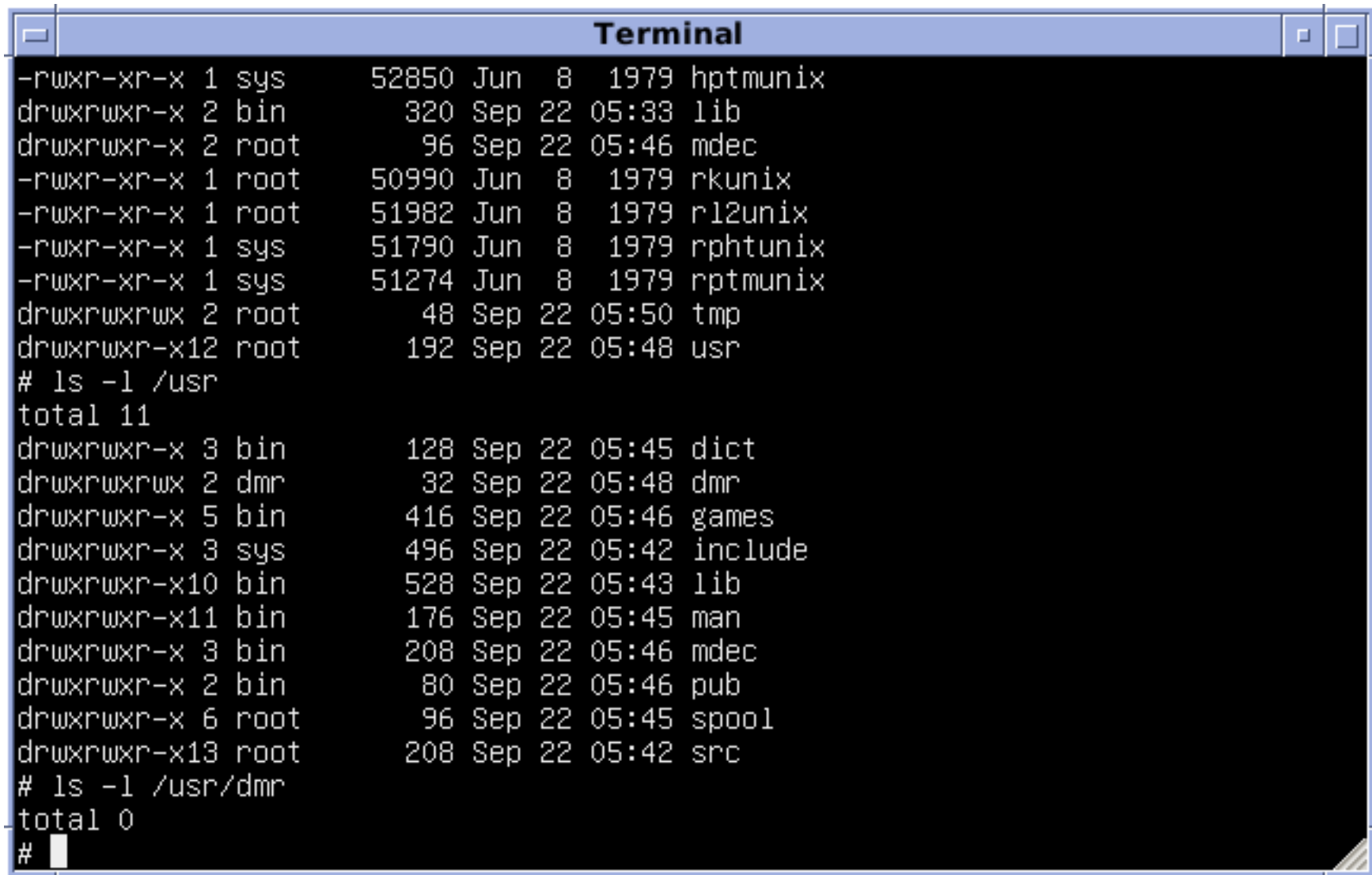
A shell is a software program
interprets and executes command lines
given by the user in consecutive lines of text

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)

A screenshot of a Unix Terminal window titled "Terminal". The window has a blue title bar and standard window controls (minimize, maximize, close) on the right. The terminal output shows a list of files with their permissions, owner, size, date, time, and name. The files are: hptmunix (permissions -rwxr-xr-x, owner 1 sys, size 52850, date Jun 8 1979), lib (permissions drwxrwxr-x, owner 2 bin, size 320, date Sep 22 05:33), mdec (permissions drwxrwxr-x, owner 2 root, size 96, date Sep 22 05:46), rkunix (permissions -rwxr-xr-x, owner 1 root, size 50990, date Jun 8 1979), rl2unix (permissions -rwxr-xr-x, owner 1 root, size 51982, date Jun 8 1979), rphtunix (permissions -rwxr-xr-x, owner 1 sys, size 51790, date Jun 8 1979), rptmunix (permissions -rwxr-xr-x, owner 1 sys, size 51274, date Jun 8 1979), tmp (permissions drwxrwxrwx, owner 2 root, size 48, date Sep 22 05:50), and usr (permissions drwxrwxr-x, owner 12 root, size 192, date Sep 22 05:48). Below this, the command "# ls -l /usr" is executed, showing a directory listing for /usr with a total size of 11. The listing includes: dict (permissions drwxrwxr-x, owner 3 bin, size 128, date Sep 22 05:45), dmr (permissions drwxrwxrwx, owner 2 dmr, size 32, date Sep 22 05:48), games (permissions drwxrwxr-x, owner 5 bin, size 416, date Sep 22 05:46), include (permissions drwxrwxr-x, owner 3 sys, size 496, date Sep 22 05:42), lib (permissions drwxrwxr-x, owner 10 bin, size 528, date Sep 22 05:43), man (permissions drwxrwxr-x, owner 11 bin, size 176, date Sep 22 05:45), mdec (permissions drwxrwxr-x, owner 3 bin, size 208, date Sep 22 05:46), pub (permissions drwxrwxr-x, owner 2 bin, size 80, date Sep 22 05:46), spool (permissions drwxrwxr-x, owner 6 root, size 96, date Sep 22 05:45), and src (permissions drwxrwxr-x, owner 13 root, size 208, date Sep 22 05:42). Finally, the command "# ls -l /usr/dmr" is executed, showing a directory listing for /usr/dmr with a total size of 0. The prompt "# " is visible at the bottom left of the terminal window.

```
-rwxr-xr-x 1 sys      52850 Jun  8  1979 hptmunix
drwxrwxr-x 2 bin        320 Sep 22  05:33 lib
drwxrwxr-x 2 root       96 Sep 22  05:46 mdec
-rwxr-xr-x 1 root    50990 Jun  8  1979 rkunix
-rwxr-xr-x 1 root    51982 Jun  8  1979 rl2unix
-rwxr-xr-x 1 sys     51790 Jun  8  1979 rphtunix
-rwxr-xr-x 1 sys     51274 Jun  8  1979 rptmunix
drwxrwxrwx 2 root       48 Sep 22  05:50 tmp
drwxrwxr-x 12 root      192 Sep 22  05:48 usr
# ls -l /usr
total 11
drwxrwxr-x 3 bin        128 Sep 22  05:45 dict
drwxrwxrwx 2 dmr         32 Sep 22  05:48 dmr
drwxrwxr-x 5 bin       416 Sep 22  05:46 games
drwxrwxr-x 3 sys       496 Sep 22  05:42 include
drwxrwxr-x 10 bin      528 Sep 22  05:43 lib
drwxrwxr-x 11 bin      176 Sep 22  05:45 man
drwxrwxr-x 3 bin      208 Sep 22  05:46 mdec
drwxrwxr-x 2 bin       80 Sep 22  05:46 pub
drwxrwxr-x 6 root       96 Sep 22  05:45 spool
drwxrwxr-x 13 root     208 Sep 22  05:42 src
# ls -l /usr/dmr
total 0
#
```

Screenshot of a Terminal application

(Source: <https://en.wikipedia.org/wiki/Unix>)

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 `-d` option of `tr`

removes a given character from the input

this case delete all carriage returns (`\r`)

Command line options can be used

in short form using a single dash (`-`)

or in a long form using two dashes (`--`)

`--delete` is equivalent to `-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
```

chmod just gave the user (u)
permissions to execute (+x)

Finally, execute the script:

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

Command line tools executed preceded by +:

```
+ 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

Single and double quotes

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*

Data Retrieval

Client Uniform Resource Locator (cURL)

a command line tool

download a URL directly into a file

List of proteins related to *caffeine*:

```
$ 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*:

```
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:

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

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

```
1 curl -s "https://www.ebi.ac.uk/chebi/viewDbAutoXrefs.  
do?d-1169080-e=1&6578706f7274=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 `gawk`
GNU implementation of `awk`

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`.
giving as input a pattern

Single and multiple patterns

Selects proteins topic CC – MISCELLANEOUS:

```
$ 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*:

```
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 `gawk`

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:

```
$ gawk -F, '{ print $1 }' < ↻  
    chebi_27732_xrefs_UniProt_relevant.csv  
  
...  
Q9VSH2  
Q15413  
Q92736
```

Comma (,) character that separates data elements

print is equivalent to echo
and \$1 the first data element

Example for first and third columns:

```
$ gawk -F, '{ print $1 ", " $3}' < ↵  
    chebi_27732_xrefs_UniProt_relevant.csv
```

...

Q9VSH2, CC - FUNCTION; CC - DISRUPTION PHENOTYPE

Q15413, CC - MISCELLANEOUS

Q92736, CC - MISCELLANEOUS

Update *getproteins.sh*:

```
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 gawk -F, '{ print $1 }'
```

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
```

Task Repetition

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
`http://www.uniprot.org/uniprot/P21817.xml`

Assembly line

Construct one URL for each protein
from the previously list
size can be large (hundreds of proteins)
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:

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

Another protein id A2AGL3 to retrieve

Another protein id B0LPN4 to retrieve

Another protein id E9PZQ0 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://www.uniprot.org/uniprot/  
  /{}.xml '
```

<https://www.uniprot.org/uniprot/A2AGL3.xml>

<https://www.uniprot.org/uniprot/B0LPN4.xml>

<https://www.uniprot.org/uniprot/E9PZQ0.xml>

...

Try these links in our internet browser

Download using the curl:

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

- 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

book file archive: <http://labs.rd.ciencias.ulisboa.pt/book/>

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 gawk -F, '{ print $1 }' | xargs -I {} curl 'https://
    www.uniprot.org/uniprot/{}.xml' -o chebi\__${ID}\_{}.
    xml
```

Includes `xargs`, `curl` and `$ID` variable
variable contains value of `$1`
avoids ambiguity `$1` in `gawk`

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.

Human proteins

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:

```
$ 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:

```
$ grep '<dbReference type="PubMed"' chebi_27732_P21817.xml ↵
```

```
<dbReference type="PubMed" id="2298749"/>
<dbReference type="PubMed" id="1354642"/>
<dbReference type="PubMed" id="8220422"/>
<dbReference type="PubMed" id="8661021"/>
<dbReference type="PubMed" id="15057824"/>
...
```

Just the identifier:

```
$ grep '<dbReference type="PubMed" ' chebi_27732_P21817.xml | gawk -F\" '{ print $4 }'
```

2298749

1354642

8220422

8661021

15057824

...

" as separation character

PubMed identifiers extraction

Apply to every protein:

```
$ grep -l '<name type="scientific">Homo sapiens</name>' ↵  
  chebi_27732_*.xml | xargs -I {} grep '<dbReference ↵  
  type="PubMed" ' {} | gawk -F\" '{ print $4 }'
```

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"' {} | gawk -F\" '{ print $4 }' | sort ↪  
  | less
```

10051009

10051009

10097181

10097181

10484775

10484775

...

`-u` option removes duplicates:

```
$ grep -l '<name type="scientific">Homo sapiens</name>' ↵  
  chebi_27732_*.xml | xargs -I {} grep '<dbReference ↵  
  type="PubMed" ' {} | gawk -F\" '{ print $4 }' | 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"' {} | gawk -F\" '{ print $4 }' | sort ↵  
  | wc
```

```
$ grep -l '<name type="scientific">Homo sapiens</name>' ↵  
  chebi_27732_*.xml | xargs -I {} grep '<dbReference ↵  
  type="PubMed"' {} | gawk -F\" '{ print $4 }' | sort ↵  
  -u | wc
```

255 255 2243

129 129 1136

`wc` prints the numbers of lines, words, and bytes
removed $255 - 129 = 126$ duplicates

```
$ expr 255 - 129
```

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 gawk -F\" '{ print $4 }' | sort -u
```

Execute:

```
$ ./getpublications.sh 27732
```

How many unique publications:

```
$ ./getpublications.sh 27732 | wc -l
```

129 as expected

Complex Elements

XML elements not in the same line

use `xmllint`

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:

```
<uniprot xmlns="http://uniprot.org/uniprot" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://uniprot.org/uniprot http://www.uniprot.org/support/docs/uniprot.xsd">
```

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 > chebi_27732_P21817_entry.↵  
    xml
```

The new XML file:

```
<entry dataset="Swiss-Prot" created="1991-05-01"  
    modified="2018-09-12" version="211">  
<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

equivalent to the previous query

specifying the full path

```
//dbReference/*
```

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"
```

```
//sequence/text()
```

the contents inside sequence

```
MGDAEGEDEVQFLRTDDEVVLQCSATVLKEQLKLCLAAEGFGNRLCFLEPTSNAQNVPPD
...
LEEHNLANYMFFLMYLINKDETEHTGQESYVWKMYQERCWDFFPAGDCFRKQYEDQLS
```

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="2298749" id="1354642" id="8220422" ...
```

Need to extract only the identifiers

Extracting XPath results

To extract the identifiers

apply `tr` command to split in multiple lines

then `gawk`:

```
$ xmllint --xpath '//dbReference[@type="PubMed"]/@id' ↵  
  chebi_27732_P21817_entry.xml | tr ' ' '\n' | gawk -F ↵  
  \" '{ NF >0 ; print $2 }'
```

`tr` replaces each space by newline character

and `gawk` extracts value inside quotes

`NF >0` select lines at least one separation character
ignores empty lines

Text Retrieval

Download the text
in titles and abstracts

UniProt citations service entry

`https://www.uniprot.org/citations/1354642`

link to the RDF/XML version

deal like XML

Retrieve the publication entry:

```
$ curl https://www.uniprot.org/citations/1354642.rdf
```

Using PubMed at NCBI:

```
$ curl 'https://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?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 '<name type="scientific">Homo sapiens</name>' ↵  
    chebi\_$_ID\_*.xml | \  
4 xargs -I {} grep '<dbReference type="PubMed"' {} | \  
5 gawk -F\" '{ print $4 }' | sort -u | \  
6 xargs -I {} curl 'https://www.uniprot.org/citations ↵  
    /{}.rdf' -o chebi\_$_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

book file archive: <http://labs.rd.ciencias.ulisboa.pt/book/>

Title and Abstract

title and abstract

values of the `title`

and `rdfs:comment` elements

Extract them:

```
$ grep -e '<title>' -e '<rdfs:comment>' ↵  
chebi_27732_1354642.rdf
```

```
<title>Polymorphisms ... hyperthermia.</title>
```

```
<rdfs:comment>Twenty-one ... gene.</rdfs:comment>
```


Remove the XML elements:

```
$ grep -e '<title>' -e '<rdfs:comment>' ↵  
    chebi_27732_1354642.rdf | gawk -F' [<>] ' '{ print $3 ↵  
    }'
```

Polymorphisms ... hyperthermia.

Twenty-one ... gene.

Two characters as field separators < and >

first field separator is <

so \$2 contains title or rdfs:comment

while \$1 is empty

second field separator is >

so \$3 contains string we want

Create *gettext.sh*:

```
1 ID=$1 # The CHEBI identifier given as input is renamed  
    to ID  
2 grep -e '<title>' -e '<rdfs:comment>' chebi\_ $ID \_*.  
    rdf | \  
3 gawk -F'[<>]' '{ print $3 }'
```

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:

```
$ 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:

```
$ 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:

```
$ 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:

```
$ grep -c -i 'malignant hyperthermia' chebi_27732.txt
$ grep -c -e 'malignant hyperthermia' -e 'Malignant ↵
hyperthermia' chebi_27732.txt
```

Output should show 96 and 95 matching lines
for the insensitive and sensitive patterns
one line 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 match:

```
$ 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

Word Matching

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'
```

- 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

Regular Expressions

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

Instead of `-e` use `-E` option

not affects matching with pattern without any operator:

```
$ echo -e 'MHS\nMHN' | grep -e 'MH'
```

```
$ echo -e 'MHS\nMHN' | grep -E 'MH'
```

Use `-e` option so `echo`

interpret `\n` as a newline.

outputs two lines

`grep` filters lines

Alternation

Alternation represented by |
either the preceding or following characters
can use parentheses specify scope

Example:

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

Basic syntax

Basic syntax no match:

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

Only if | and parentheses are in string:

```
$ echo -e 'MH(S|N)' | grep -e 'MH(S|N)'
```

Scope

Remove the parentheses and add `-w`:

```
$ echo -e 'MHS\nMHN' | grep -w -E 'MHS|N'
```

Only first line since operator applied to all preceding characters

If add a single `N` get another match:

```
$ echo -e 'MHS\nN' | grep -w -E 'MHS|N'
```

Move the opening parenthesis:

```
$ echo -e 'MHS\nMHN' | grep -E 'M(HS|N)'
```

Only *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 96 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)

...

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
```

154 matches were found
in 45 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:

```
$ 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]'
```

Keep case sensitive grep:

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

Dot character inside a range represents itself:

```
$ echo -e 'MHS\nMH.' | grep -E '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

Quantifiers

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 *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:

```
$ 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:

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

```
$ 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

Position

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 \$

Beginning

Lines starting with *Malignant Hyperthermia*:

```
$ 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:

```
$ 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 22 of the 96 matches were considered

Ending

Lines ending with *Malignant Hyperthermia*,

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

```
Novel mutation in the RYR1 gene (R2454C) in a patient  
with malignant hyperthermia.
```

```
Identification of a novel mutation in the ryanodine  
receptor gene (RYR1) in patients with malignant  
hyperthermia.
```

```
Novel skeletal muscle ryanodine receptor mutation in a  
large Brazilian family with 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 96 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.

...

Full line

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 `-b`

```
$ echo 'MHS MHN MHE' | grep -b -o -w -E 'MH[SNE]'
```

```
0:MHS
```

```
4:MHN
```

```
8:MHE
```

Same result in multiple lines:

```
$ echo -e 'MHS\nMHN\nMHE' | grep -b -o -w -E 'MH[SNE]'
```

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 1493 lines from the original 248 lines:

```
$ wc -l chebi_27732.txt
```

Wrong tokens

Not so simple, analyze:

```
$ 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:

```
$ 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 1067 lines from original 248 lines:

```
$ 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


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

From:

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

To:

```
... muscle relaxants.
```

```
To date, ...
```

Sentences starting after delimiter without space between:

```
... bulk.Fetal ...  
... sequencing.Whole ...
```

Include delimiter directly followed by letter:

```
$ sed -E 's/([.!?])( +)/\1\n\2/g' chebi_27732.txt | grep  
-i '[.!?][a-z]'
```

Compulsory space become optional
but requiring uppercase letter:

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

Get 1127 lines,
able to split more 60 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 grep -e '<title>' -e '<rdfs:comment>' chebi\_ $ID \_*.  
    rdf | \  
3 gawk -F' [<>] ' '{ print $3 }' | \  
4 sed -E 's/([.!?])( *[A-Z])/\\1\\n\\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 `-n` get the number of line:

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

...

1107:MH

1107:MH

1109:MH

1111:MH

1112:MH

Add the `-b` option:

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

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

...

1107:162025:MH

1107:162145:MH

1109:162435:MH

1111:162867:MH

1112:163193:MH

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*:

```
1 PATTERN=$1
2 grep -n -o -w -E $PATTERN | \
3 tr ':' '\t'
```

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

Execute:

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

...

1107 MH

1107 MH

1109 MH

1111 MH

1112 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

```
$ 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
```

```
...
```

```
1110:MH
```

```
1110:caffeine
```

```
1111:caffeine
```

```
1111: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
```

```
239: ... MHS families were investigated with a  
caffeine ...
```

One of the seven displayed sentences

Missing all with *caffeine* first:

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

```
801: ... caffeine-halothane contracture test were  
greater in those who had a known MH ...
```

```
1111: ... 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'
```

Show nine 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:

50: 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

Ignoring specific type of relations

use `-v`:

```
$ 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

OWL files

Retrieving both ontologies (OWL files):

```
$ curl -O 'https://raw.githubusercontent.com/  
DiseaseOntology/HumanDiseaseOntology/master/src/  
ontology/releases/2018-11-02/doid.owl'  
$ curl -O 'ftp://ftp.ebi.ac.uk/pub/databases/chebi/  
archive/rel169/ontology/chebi_lite.owl'
```

-O saves to file name as remote file (last part of URL)
files *chebi_lite.owl* and *doid.owl*

Links for specific releases

another release may change the examples output
check links on BioPortal or OBO Foundry
or from book file archive

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>
  <obo:IAO_0000115
```

```
...
<oboInOwl:hasDbXref rdf:datatype="http://www.w3
.org/2001/XMLSchema#string">UMLS_CUI:C0024591<
/oboInOwl:hasDbXref>
<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>
<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_MGI_slim"/>
```

```
<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: malignant hyperthermia

Term IRI: http://purl.obolibrary.org/obo/DOID_8545

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. [database_cross_reference: [url:http://en.wikipedia.org/wiki/Malignant_hyperthermia](http://en.wikipedia.org/wiki/Malignant_hyperthermia)][database_cross_reference: [url:http://en.wikipedia.org/wiki/Malignant_hyperthermia](http://en.wikipedia.org/wiki/Malignant_hyperthermia)][database_cross_reference: [url:http://en.wikipedia.org/wiki/Malignant_hyperthermia](http://en.wikipedia.org/wiki/Malignant_hyperthermia)][database_cross_reference: [url:http://en.wikipedia.org/wiki/Malignant_hyperthermia](http://en.wikipedia.org/wiki/Malignant_hyperthermia)]

Annotations

- **database_cross_reference:** ICD9CM:995.86; MESH:D008305; ICD10CM:T88.3; UMLS_CUI:C0024591; ORDO:423; CSP2005:2871-4352; GARD:6964; MTHICD9_2006:995.86; NCI:C84869; OMIM:PS145600
- **has_exact_synonym:** anesthesia related hyperthermia; malignant hyperpyrexia due to anesthesia
- **has_obo_namespace:** disease_ontology
- **http://www.w3.org/2000/01/rdf-schema#comment:** Xref MGI. OMIM mapping confirmed by DO. [SN].
- **id:** DOID:8545
- **in_subset:** DO MGI slim; DO rare slim; NCItthesaurus

Class Hierarchy

```

Thing
+ disease
+ disease of anatomical entity
+ musculoskeletal system disease
+ muscular disease
+ muscle tissue disease
- distal arthrogryposis
- rippling muscle disease 2
- rippling muscle disease 1
- myostatin-related muscle hypertrophy
- myotonia congenita
+ myopathy
- malignant hyperthermia

```

Class description of *malignant hyperthermia* in the Human Disease Ontology

(Source: <http://www.ontobee.org/>)

malignant hyperthermia subclass of (specialization)

entries 0050736

and 66 *muscle tissue disease*

malignant hyperthermia a special case of *muscle tissue disease*

Retrieve full class definition of *caffeine*:

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

Semantics of *caffeine* differs from *malignant hyperthermia*
still share many properties
such as `subClassOf`

```
<owl:Class rdf:about="http://purl.obolibrary.org/obo/
  CHEBI_27732">
  <rdfs:subClassOf rdf:resource="http://purl.
    obolibrary.org/obo/CHEBI_26385"/>
  <rdfs:subClassOf rdf:resource="http://purl.
    obolibrary.org/obo/CHEBI_27134"/>
  <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>
  ...
  <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_85234"/>
</owl:Restriction>
</rdfs:subClassOf>
<obo:IAO_0000115 rdf:datatype="http://www.w3.org
  /2001/XMLSchema#string">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.</
  obo:IAO_0000115>
<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>
```

Class: caffeine

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

Definition: 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.

Annotations

- **database_cross_reference:** PMID:15257305; PMID:10822912; PMID:18421070; PMID:16528931; PMID:22770225; PMID:12943586; PMID:17957400; PMID:8679661; PMID:12397877; KNApSack:C00001492; PMID:14521986; PMID:11815511; PMID:11431501; PMID:20164568; Beilstein:17705; PMID:11209966; PMID:9132918; PMID:11410911; PMID:16709440; PMID:11014293; PMID:18625110; Gmelin:103040; MetaCyc:1-3-7-TRIMETHYLXANTHINE; PMID:19879252; KEGG:C07481; PMID:12457274; PMID:10803761; PMID:19088793; HMDB:HMDB0001847; PMID:7689104; PMID:14607010; KEGG:D00528; PMID:16143823; PMID:11949272; DrugBank:DB00201; PMID:15280431; PMID:10884512; PMID:17387608; PMID:16856769; PMID:19084078; PMID:16644114; PMID:10924888; PMID:10796597; PMID:11022879; LINC:LSM-2026; PMID:10510174; PMID:16805851; PMID:8347173; PDBeChem:CFF; PMID:7441110; PMID:16391865; PMID:9218278; PMID:15840517; PMID:9067318; PMID:18258404; Drug_Central:463; PMID:19418355; PMID:17508167; PMID:17724925; PMID:12574990; PMID:10983026; PMID:15718055; Reaxys:17705; PMID:19007524; Wikipedia:Caffeine; PMID:9063686; PMID:18647558; PMID:18068204; CAS:58-08-2; PMID:17132260; PMID:20470411; PMID:8332255; PMID:11312039; PMID:15681408; PMID:17932622; PMID:19047957; PMID:12915014
- **has_alternative_id:** CHEBI:22982; CHEBI:41472; CHEBI:3295
- **has_exact_synonym:** CAFFEINE; Caffeine; 1,3,7-trimethyl-3,7-dihydro-1H-purine-2,6-dione; caffeine
- **has_obo_namespace:** chebi_ontology
- **has_related_synonym:** Thein; guaranine; cafeine; theine; 1-methyltheobromine; 1,3,7-trimethyl-2,6-dioxopurine; 3,7-Dihydro-1,3,7-trimethyl-1H-purin-2,6-dion; 1,3,7-trimethylxanthine; anhydrous caffeine; 1,3,7-Trimethylxanthine; 7-methyltheophylline; Coffein; cafeina; 1,3,7-trimethylpurine-2,6-dione; mateina; methyltheobromine; Koffein; teina
- **http://purl.obolibrary.org/obo/chebi/charge:** 0
- **http://purl.obolibrary.org/obo/chebi/formula:** C8H10N4O2
- **http://purl.obolibrary.org/obo/chebi/inchi:** InChI=1S/C8H10N4O2/c1-10-4-9-6-5(10)7(13)12(3)8(14)11(6)2/h4H,1-3H3
- **http://purl.obolibrary.org/obo/chebi/inchikey:** RYYVLZVUVJVGH-UHFFFAOYSA-N
- **http://purl.obolibrary.org/obo/chebi/mass:** 194.19076
- **http://purl.obolibrary.org/obo/chebi/monoisotopicmass:** 194.080
- **http://purl.obolibrary.org/obo/chebi/smiles:** Cn1cnc2n(C)c(=O)n(C)c(=O)c12
- **http://www.geneontology.org/formats/oboInOwl#id:** CHEBI:27732
- **in_subset:** http://purl.obolibrary.org/obo/chebi#3_STAR

Class Hierarchy

```

Thing
+ chemical entity
  + molecular entity
    + main group molecular entity
      + p-block molecular entity
        + carbon group molecular entity
          + organic molecular entity
            + organic molecule
              + organic cyclic compound
                + organic heterocyclic compound
                  + organic heteropolycyclic compound
                    + organic heterobicyclic compound
                      + imidazopyrimidine
                        + purines
                          + purine alkaloid
                            + methylxanthine
                              + trimethylxanthine
                                - 8-(3-chlorostyryl)caffeine
                                - caffeine

```

Class description of *caffeine* in ChEBI

(Source: <http://www.ontobee.org/>)

caffeine specialization of
26385 *purine alkaloid* and 27134 *trimethylxanthine*

Have additional subclass relationships
not subsumption (*is-a*).

Related Classes

Superclasses & Asserted Axioms

- [muscle tissue disease](#)
- [autosomal dominant disease](#)
- [has material basis in](#) some [autosomal dominant inheritance](#)

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

(Source: <http://www.ontobee.org/>)

Superclasses & Asserted Axioms

- [has role](#) some [human blood serum metabolite](#)
- [has role](#) some [mouse metabolite](#)
- [has role](#) some [plant metabolite](#)
- [has role](#) some [fungal metabolite](#)
- [has role](#) some [environmental contaminant](#)
- [has role](#) some [adjuvant](#)
- [has role](#) some [food additive](#)
- [has role](#) some [ryanodine receptor agonist](#)
- [has role](#) some [adenosine receptor antagonist](#)
- [has role](#) some [ryanodine receptor modulator](#)
- [has role](#) some [EC 3.1.4.* \(phosphoric diester hydrolase\) 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 [central nervous system stimulant](#)
- [has role](#) some [psychotropic drug](#)
- [has role](#) some [diuretic](#)
- [has role](#) some [xenobiotic](#)
- [has role](#) some [mutagen](#)
- [purine alkaloid](#)
- [trimethylxanthine](#)

Related classes of *caffeine* in ChEBI

(Source: <http://www.ontobee.org/>)

relationship between *caffeine* and
the entry 25435 *mutagen*
defined by 0000087 *has role*
of the *Relations Ontology*.

Means *caffeine has role mutagen*

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

Annotations

- database_cross_reference: RO:0000087
- has_obo_namespace: chebi_ontology
- [http://www.geneontology.org/formats/oboInOwl#id: has_role](http://www.geneontology.org/formats/oboInOwl#id:has_role)
- [http://www.geneontology.org/formats/oboInOwl#is_cyclic: false](http://www.geneontology.org/formats/oboInOwl#is_cyclic:false)
- [http://www.geneontology.org/formats/oboInOwl#is_transitive: false](http://www.geneontology.org/formats/oboInOwl#is_transitive:false)
- shorthand: has_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*:

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

```
rdf:about="http://purl.obolibrary.org/obo/DOID_8545"
```

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

extracts the URI specified
as an attribute of that class.

Only the value, add `string`:

```
$ xmllint --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

Get URI of *caffeine*:

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


Script *geturi.sh*:

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

Multiple labels as standard input

OWL file to find URIs as argument

xargs process each line of standard input

tr because xmllint same line,

split using " for URI

grep keep only URIs

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']='http://purl.obolibrary.org/obo/DOID_8545']/*[local-name()='label']/text()" doid.owl  
  
malignant hyperthermia
```

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

Get label of compound 27732:

```
$ xmllint --xpath "//*[local-name()='Class'][@*[local-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']" $OWLFILE | \
3 tr '<>' '\n' | \
4 grep -v -e ':label' -e '^$'

```

Multiple URIs as standard input

OWL file to find labels as argument

`xargs` process each line of standard input

`text` not adds newline after each match

split in multiple lines using `tr`

filtering `:label` keyword or are empty `^$`

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\  
nhttp://purl.obolibrary.org/obo/DOID_66' | ./getlabels.sh doid.owl  
  
$ echo -e 'http://purl.obolibrary.org/obo/CHEBI_27732\  
nhttp://purl.obolibrary.org/obo/CHEBI_26385\  
nhttp://purl.obolibrary.org/obo/CHEBI_27134' | ./getlabels.sh chebi_lite.owl
```

malignant hyperthermia

muscle tissue disease

caffeine

purine alkaloid

trimethylxanthine

Test both scripts:

```
$ echo -e 'malignant hyperthermia\nmuscle tissue disease'  
      ' | ./geturi.sh doid.owl | ./getlabels.sh doid.owl  
$ echo -e 'caffeine\npurine alkaloid\ntrimethylxanthine'  
      | ./geturi.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\  
nhttp://purl.obolibrary.org/obo/DOID_66' | ./getlabels.sh doid.owl | ./geturi.sh doid.owl  
$ echo -e 'http://purl.obolibrary.org/obo/CHEBI_27732\  
nhttp://purl.obolibrary.org/obo/CHEBI_26385\  
nhttp://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:

```
1 xmllint --xpath "//*[local-name()='Class'][@*[local-name()='about']='http://purl.obolibrary.org/obo/DOID_8545']/*[local-name()='hasExactSynonym' or local-name()='label']" 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>
```

```
<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']" $OWLFILE | \
3 tr '<>' '\n' | \
4 grep -v -e ':label' -e ':hasExactSynonym' -e '
    hasRelatedSynonym' -e '^$'
```

Adding the hasExactSynonym keyword and hasRelatedSynonym

Execute:

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

anesthesia related hyperthermia

malignant hyperpyrexia due to anesthesia

malignant hyperthermia

URI of synonyms

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:

```
$ 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:

```
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 tr '"' '\n' | grep 'http'
```


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
```

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

Parent classes of *malignant hyperthermia*:

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

```
$ 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:

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

http://purl.obolibrary.org/obo/CHEBI_26385

http://purl.obolibrary.org/obo/CHEBI_27134

Script *getparents.sh*:

```
1 OWLFILE=$1
2 xargs -I {} xmllint --xpath "//*[local-name()='Class'
    '][@*[local-name()='about']='{}']/*[local-name()='
    subClassOf']/@*[local-name()='resource']" $OWLFILE |
    \
3 tr '"' '\n' | grep 'http'
```

Multiple URIs given as standard input
OWL file to find parents as argument

Parents of *malignant hyperthermia*:

```
$ 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:

```
$ 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*:

```
$ 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 *subClassOf*:

```
$ 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 | tr '"' '\n' | grep 'http'
```

Related classes are

in attribute *resource*

of *someValuesFrom* element

inside *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_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*:

```
$ 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 | tr '"' '\n' | grep 'http' | ./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

Ancestors

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*:

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

Recursion

Script *getancestors.sh*:

```
1 OWLFILE=$1
2 CLASSES=$(cat -)
3 [[ -z "$CLASSES" ]] && exit
4 PARENTS=$(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 `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 `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 ↵
8         $OWLFILE | sort -u)
9     ANCESTORS="$ANCESTORS\n$PARENTS"
10    CLASSES=$PARENTS
11 done
12 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*:

```
$ 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:

```
$ 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*:

```
$ 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

Ancestors 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*:

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

Check contents:

```
$ cat chebi_27732_lexicon.txt | sort -u
alkaloid
aromatic compound
bicyclic compound
caffeine
...
```

This lexicon is much larger.

Merging labels

Merging two lexicons in *lexicon.txt*:

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

Recognize any mention in *chebi_27732_sentences.txt*:

```
$ 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*:
add 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

Generic Lexicon

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']" doid.owl
```

Script *getalllabels.sh*:

```
1 OWLFILE=$1
2 xmllint --xpath "//*[local-name()='Class']/*[local-name()='hasExactSynonym' or local-name()='hasRelatedSynonym' or local-name()='label']" \
   $OWLFILE | \
3 tr '<>' '\n' | \
4 grep -v -e ':label' -e ':hasExactSynonym' -e 'hasRelatedSynonym' -e '^$' | \
5 sort -u
```

Execute:

```
$ ./getalllabels.sh doid.owl
```

```
11-beta-hydroxysteroid dehydrogenase deficiency type 2
```

```
11p partial monosomy syndrome
```

```
1,4-phenylenediamine allergic contact dermatitis
```

```
...
```

```
Zoophilia
```

```
Zoophobia
```

```
zygomycosis
```

Redirect to *diseases.txt*:

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

Check how many labels:

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

More than 29 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

& 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:

```
$ grep -c -F '(' diseases.txt  
$ grep -c -F ',' diseases.txt  
$ grep -c -F '[' diseases.txt  
$ grep -c -F ':' diseases.txt  
$ grep -c -F '&' 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, nondelation 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:

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

Enforces a space after the separation char

avoids: *46,XY DSD due to LHB deficiency*

Recognize both *ATR* and *ATR syndrome*:

```
$ tr -d '[](){}' < diseases.txt | sed -E 's/[,:;] .*$//'  
| 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 OWLFILE=$1
2 xmllint --xpath "//*[local-name()='Class']/*[local-name()='hasExactSynonym' or local-name()='hasRelatedSynonym' or local-name()='label']" $OWLFILE | \
3 tr '<>' '\n' | \
4 grep -v -e ':label' -e ':hasExactSynonym' -e 'hasRelatedSynonym' -e '^$' | \
5 tr -d '[](){}' | \
6 sed -E 's/[, :;] .*$/;/ s/^ *//; s/ *$//' | sort -u
```

Generate fixed lexicon:

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

Check number of entries:

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

About 28 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
```

43 diseases related *caffeine*:

Andersen-Tawil syndrome

arrhythmogenic right ventricular cardiomyopathy

ARVD2

ataxia telangiectasia

ATR

atrial fibrillation

benign congenital myopathy

cancer

cardiac arrest

cardiomyopathy

catecholaminergic polymorphic ventricular tachycardia

central core disease

chorea

congenital hip dislocation

congenital myopathy

deficiency

disease

dystonia

epilepsy

FHL1

hand

hepatitis C

HL

hypercholesterolaemia

hypokalemic periodic paralysis

Hypokalemic periodic paralysis

intellectual disability

long QT syndrome

LQT1

LQT2

LQT3

LQT5

LQT6

malignant hyperthermia

migraine

myopathy

myotonic dystrophy type 1

nemaline myopathy

nemaline rod myopathy
ophthalmoplegia
rod myopathy
scoliosis
syndrome

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
more than 100 times faster

Case insensitive

Performance issue

with `-i`:

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

Solution convert both lexicon and text

to lowercase (or uppercase)

problem incorrectly matching acronyms in lowercase

ASCII encoding

Case insensitive matching issue
due to UTF-8 character encoding
instead of ASCII

UTF-8 allow special characters
euro symbol in standard way

Normal text without special characters
ASCII works fine and efficiently

Unix shells specify usage of ASCII encoding
by `LC_ALL=C` before command

Execute:

```
$ LC_ALL=C grep -n -o -w -F -i -f diseases.txt ↵  
chebi_27732_sentences.txt
```

Significant increase in performance

Check how many labels recognized:

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

60 labels being recognized.

Check new labels recognized:

```
$ LC_ALL=C 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
```

Arrhythmogenic right ventricular dysplasia

arthrogryposis

can

Catecholaminergic polymorphic ventricular tachycardia

Central Core Disease

defect

Disease

dyskinesia

face

fever

Malignant hyperthermia

Malignant Hyperthermia

March

ORF

total

Correct matches

Some only recognized by case insensitive match
arthrogryposis

Lexicon not include lowercase case:

```
$ grep -i '^arthrogryposis$' diseases.txt
```

Arthrogryposis

ARTHROGRYPOSIS

Check in text:

```
$ grep -w -i 'arthrogryposis' chebi_27732_sentences.txt
```

Only lowercase:

```
... (multiple arthrogryposis, congenital dislocation  
    of the hips ...  
... fetal akinesia, arthrogryposis multiplex ...
```


Another example:

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

Lexicon only name with first character in uppercase:

Dyskinesia

Incorrect matches

Case insensitive match create other problems

CAN for *Crouzon syndrome-acanthosis nigricans syndrome*:

```
$ grep -i '^CAN$' diseases.txt
```

Check how many times *CAN* is recognized:

```
$ LC_ALL=C grep -n -o -w -i -F -f diseases.txt ↵  
chebi_27732_sentences.txt | grep -i ':CAN' | wc -l
```

18 times

Which type of matches:

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

Incorrect mentions:

can

18 mismatches by case insensitive match

Entity Linking

What recognized labels represent

Find what *AD2* represents:

```
$ echo "AD2" | ./geturi.sh doid.owl | ./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

Ambiguity

Classes acronym *ATS* may represent:

```
$ echo "ATS" | ./geturi.sh doid.owl
```

```
http://purl.obolibrary.org/obo/DOID_0050434
```

```
http://purl.obolibrary.org/obo/DOID_0110034
```

Two distinct diseases:

Andersen-Tawil syndrome

X-linked Alport syndrome

Alternative labels:

```
$ echo "http://purl.obolibrary.org/obo/DOID_0050434" | ↩  
    ./getlabels.sh doid.owl  
$ echo "http://purl.obolibrary.org/obo/DOID_0110034" | ↩  
    ./getlabels.sh doid.owl
```

Both containing *ATS* as expected:

```
ANDERSEN CARDIODYSRHYTHMIC PERIODIC PARALYSIS  
ATS  
Andersen syndrome  
LQT7  
Long QT syndrome 7  
Potassium-Sensitive Cardiodysrhythmic Type  
Andersen-Tawil syndrome  
  
ATS  
nephropathy and deafness, X-linked  
X-linked Alport syndrome
```

Surrounding entities

Select class closer in meaning
to other classes in surrounding text

Assuming entities in same text
semantically related

Example:

... channel genes, KCNQ1 (LQT1), KCNH2 (LQT2), SCN5A (LQT3), KCNE1 (LQT5), and KCNE2 (LQT6), along with KCNJ2 (Andersen-Tawil syndrome) and ...

Replace *Andersen-Tawil syndrome* by *ATS*:

... channel genes, KCNQ1 (LQT1), KCNH2 (LQT2), SCN5A (LQT3), KCNE1 (LQT5), and KCNE2 (LQT6), along with KCNJ2 (ATS) and ...

Identify the diseases:

```
$ echo 'channel genes, KCNQ1 (LQT1), KCNH2 (LQT2), SCN5A  
      (LQT3), KCNE1 (LQT5), and KCNE2 (LQT6), along with  
      KCNJ2 (ATS) and' | grep -o -w -F -f diseases.txt
```

LQT1

LQT2

LQT3

LQT5

LQT6

ATS

Find URIs:

```
$ echo 'channel genes, KCNQ1 (LQT1), KCNH2 (LQT2), SCN5A  
    (LQT3), KCNE1 (LQT5), and KCNE2 (LQT6), along with  
    KCNJ2 (ATS) and' | grep -o -w -F -f diseases.txt |  
    ./geturi.sh doid.owl
```

Ambiguity for *ATS*

Andersen-Tawil syndrome (DOID:0050434)

X-linked Alport syndrome (DOID:0110034):

http://purl.obolibrary.org/obo/DOID_0110644

http://purl.obolibrary.org/obo/DOID_0110645

http://purl.obolibrary.org/obo/DOID_0110646

http://purl.obolibrary.org/obo/DOID_0110647

http://purl.obolibrary.org/obo/DOID_0110648

http://purl.obolibrary.org/obo/DOID_0050434

http://purl.obolibrary.org/obo/DOID_0110034

Semantic similarity

Solve ambiguity problems

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

Web tool DiShIn

calculate semantic similarity between:

LQT1 (DOID:0110644) and *Andersen-Tawil syndrome* (DOID:0050434)
and *LQT1* and *X-linked Alport syndrome* (DOID:0110034)

The screenshot shows a web browser window with the URL `labs.rd.ciencias.ulisboa.pt/dishin/`. The page title is "DiShIn: Semantic Similarity Measures using Disjunctive Shared Information". It features a dropdown menu for "Ontology" set to "DO - Human Disease Ontology". Two input fields, "Entry 1" and "Entry 2", contain the DOIDs `DOID:0110644` and `DOID:0050434` respectively. Below each entry are example strings of identifiers. A "Submit" button is located below the second entry. At the bottom, a table displays similarity results for different measures and methods.

Measure	MICA/DiShIn	(Ex/In)trinsic	Similarity
Resnik	DiShIn	intrinsic	3.1715006566
Resnik	MICA	intrinsic	6.34300131319
Lin	DiShIn	intrinsic	0.376553538118
Lin	MICA	intrinsic	0.753107076235
JC	DiShIn	intrinsic	0.0952210062728
JC	MICA	intrinsic	0.240449173481

Semantic similarity between *LQT1* (DOID:0110644)
and *Andersen-Tawil syndrome* (DOID:0050434)

The screenshot shows a web browser window with the URL `labs.rd.ciencias.ulisboa.pt/dishin/`. The page title is "DiShIn: Semantic Similarity Measures using Disjunctive Shared Information".

Ontology
 DO - Human Disease Ontology

Entry 1

Examples: CHEBI:31236, DOID:2841, GO:0000023 (or protein Q12345), HP:0000588, gold, RID16139, or ambulance-noun-1

Entry 2

Examples: CHEBI:3131, DOID:1324, GO:0000025 (or protein Q12346), HP:0001093, copper, RID16140, or motorcycle-noun-1

Measure	MICA/DiShIn	(Ex/In)trinsic	Similarity
Resnik	DiShIn	intrinsic	0.0
Resnik	MICA	intrinsic	0.0
Lin	DiShIn	intrinsic	0.0
Lin	MICA	intrinsic	-0.0
JC	DiShIn	intrinsic	0.0593651994576
JC	MICA	intrinsic	0.0593651994576

Semantic similarity between *LQT1* (DOID:0110644)
 and *X-linked Alport syndrome* (DOID:0110034)

Measures

DiShIn provides three measures

Resnik, Lin and Jiang-Conrath

last two values between 0 and 1,

Jiang-Conrath distance converted similarity

LQT1 more similar to *Andersen-Tawil syndrome*
than to *X-linked Alport syndrome*

Similar results if replace
LQT1 by *LQT2*, *LQT3*, *LQT5*, or *LQT6*.

Semantic similarity can identify
Andersen-Tawil syndrome correct linked entity
for *ATS* in this text

DiShIn installation

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

Install it locally:

```
$ git clone git://github.com/lasigeBioTM/DiShIn
```

Copy Human Disease Ontology:

```
$ cp doid.owl DiShIn/
```

```
$ cd DiShIn
```

Database file

Convert *doid.owl* into database *doid.db*:

```
$ python dishin.py doid.owl doid.db http://purl.↵  
  obolibrary.org/obo/ http://www.w3.org/2000/01/rdf-↵  
  schema#subClassOf ''
```

Alternatively, download latest database version:

```
$ curl -O http://labs.rd.ciencias.ulisboa.pt/book/doid.↵  
  db
```

DiShIn execution

Execute:

```
$ python dishin.py doid.db DOID_0110644 DOID_0050434
```

```
$ python dishin.py doid.db DOID_0110644 DOID_0110034
```

Values between *LQT1* (DOID:0110644)
and *Andersen-Tawil syndrome* (DOID:0050434):

```
Resnik DiShIn intrinsic 3.1715006566
Resnik MICA intrinsic 6.34300131319
Lin DiShIn intrinsic 0.376553538118
Lin MICA intrinsic 0.753107076235
JC DiShIn intrinsic 0.0952210062728
JC MICA intrinsic 0.240449173481
```

Values between *LQT1* (DOID:0110644)
and *X-linked Alport syndrome* (DOID:0110034):

```
Resnik DiShIn intrinsic 0.0
Resnik MICA intrinsic 0.0
Lin DiShIn intrinsic 0.0
Lin MICA intrinsic -0.0
JC DiShIn intrinsic 0.0593651994576
JC MICA intrinsic 0.0593651994576
```

Return to parent folder:

```
$ cd ..
```

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:

```
$ git clone git://github.com/lasigeBioTM/MER
```

Copy Human Disease Ontology:

```
$ cp doid.owl MER/data/
```

```
$ cd MER
```

Lexicon files

Create lexicon:

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

Alternatively, download latest lexicon version:

```
$ curl -O http://labs.rd.ciencias.ulisboa.pt/book/  
    doid_lexicons.zip  
$ unzip doid_lexicons.zip -d data/
```

Check the contents:

```
$ tail data/doid*
```



```
==> data/doid_links.tsv <==  
zika virus disease http://purl.obolibrary.org/obo/  
  DOID_0060478  
zikv congenital infection http://purl.obolibrary.org/  
  obo/DOID_0080180  
zinacef allergy http://purl.obolibrary.org/obo/  
  DOID_0040025  
zinsser-cole-engman syndrome http://purl.obolibrary.  
  org/obo/DOID_0070025  
ziziphus mauritiana fruit allergy http://purl.  
  obolibrary.org/obo/DOID_0060507  
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
```

zygomycosis http://purl.obolibrary.org/obo/DOID_8485

==> data/doid.txt <==

zika virus disease

zikk congenital infection

zinacef allergy

zinsse-ole-engman syndrome

ziziphus mauritiana fruit allergy

zlotogora-zilberman-tenenbaum syndrome

zollinger-ellison syndrome

zoophilia

zoophobia

zygomycosis

==> data/doid_word1.txt <==

xph

xpid

xpv
xscid
yaba
yaws
zaspopathy
zoophilia
zoophobia
zygomycosis

=> data/doid_word2.txt <=
yunis.varon syndrome
zantac allergy
zebrafish allergy
zellweger syndrome
zemuron allergy
zika fever
zinacef allergy

zinsser.cole.engman syndrome
zlotogora.zilberman.tenenbaum syndrome
zollinger.ellison syndrome

==> data/doid_words2.txt <==

yersinia infectious
yersinia pestis
yersinia pseudotuberculosis
y.linked monogenic
y.linked sertoli
y.linked spermatogenic
yolk sac
zika virus
zikv congenital
ziziphus mauritiana

==> data/doid_words.txt <==

y.linked spermatogenic failure 1
y.linked spermatogenic failure 2
yolk sac neoplasm
yolk sac tumor
yolk sac tumor of mediastinum
yolk sac tumor of the cns
zika virus congenital syndrome
zika virus disease
zikv congenital infection
ziziphus mauritiana fruit allergy

MER execution

Execute MER:

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

Removed 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
```

	A	B	C	D
1	89	111	<u>malignant hyperthermia</u>	<u>http://purl.obolibrary.org/obo/DOID_8545</u>
2	74	96	<u>malignant hyperthermia</u>	<u>http://purl.obolibrary.org/obo/DOID_8545</u>
3	157	164	<u>disease</u>	<u>http://purl.obolibrary.org/obo/DOID_4</u>
4	144	164	<u>central core disease</u>	<u>http://purl.obolibrary.org/obo/DOID_3529</u>
5	13	20	<u>disease</u>	<u>http://purl.obolibrary.org/obo/DOID_4</u>
6	47	55	<u>myopathy</u>	<u>http://purl.obolibrary.org/obo/DOID_423</u>

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